RM0514
Reference manual

STM32G56x/55x/54x/53x Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32G56x/55x/54x/53x microcontrollers memory and peripherals.

For ordering information, mechanical and electrical device characteristics, refer to the corresponding datasheets.

For information on the Arm® Cortex®-M33 core, refer to the corresponding Arm® Technical Reference Manual available on http://infocenter.arm.com.

STM32G56x/55x/54x/53x microcontrollers include ST state-of-the-art patented technology.

Related documents

- STM32G56x/55x/54x/53x datasheets (DS14920, DS14970)
- STM32G56x/55x/54x/53x device errata (ES0656, ES0657)
- STM32 Cortex®-M33 MCUs programming manual (PM0264)

Contents

1 Documentation conventions 111

1.1 General information 111

1.2 List of abbreviations for registers 111

1.3 Register reset value 112

1.4 Glossary 112

1.5 Availability of peripherals 112

2 Memory and bus architecture 113

2.1 System architecture 113

2.1.1 Fast C-bus 115

2.1.2 Slow C-bus 115

2.1.3 S-bus 115

2.1.4 DCACHE S-bus 116

2.1.5 GPDMA1 and GPDMA2 buses 116

2.1.6 BusMatrix 116

2.1.7 AHB/APB bridges 116

2.2 TrustZone security architecture 116

2.2.1 Default TrustZone security state 117

2.2.2 TrustZone peripheral classification 118

2.3 Memory organization 122

2.3.1 Introduction 122

2.3.2 Memory map and register boundary addresses 123

2.3.3 Embedded SRAM 129

2.3.4 Flash memory overview 130

2.3.5 Boot modes 130

3 System security 131

3.1 Key security features 131

3.2 Secure install 132

3.3 Secure boot 133

3.3.1 Unique boot entry 133

3.3.2 Immutable root of trust in system flash memory 134

3.3.3 Temporal isolation levels 134

3.4 Secure update . . . 134
3.4.1 Live update . . . 135
3.5 Resource isolation using TrustZone . . . 135
3.5.1 TrustZone security architecture . . . 136
3.5.2 Armv8-M security extension of Cortex-M33 . . . 136
3.5.3 Memory and peripheral allocation using IDAU/SAU . . . 136
3.5.4 Memory and peripheral allocation using GTZC . . . 138
3.5.5 Managing security in TrustZone-aware peripherals . . . 142
3.5.6 Activating TrustZone security . . . 148
3.5.7 Deactivating TrustZone security . . . 149
3.5.8 Resource isolation using Cortex privileged mode . . . 149
3.6 Secure execution . . . 153
3.6.1 Memory protection unit (MPU) . . . 153
3.6.2 Embedded flash memory write protection . . . 154
3.6.3 Tamper detection and response . . . 154
3.7 Secure storage . . . 156
3.8 Product life cycle . . . 157
3.8.1 Life-cycle management . . . 159
3.8.2 Recommended product settings . . . 162
3.8.3 Software intellectual property protection . . . 162
3.8.4 Software intellectual property protection with OTFDEC . . . 162
3.8.5 Other software intellectual property protections . . . 164

4 Boot modes . . . 165

5 Global TrustZone® controller (GTZC) . . . 168
5.1 GTZC introduction . . . 168
5.2 GTZC main features . . . 168
5.3 GTZC implementation . . . 170
5.4 GTZC functional description . . . 172
5.4.1 GTZC block diagram . . . 172
5.4.2 Illegal access definition . . . 173
5.4.3 TrustZone security controller (TZSC) . . . 173
5.4.4 Memory protection controller - block based (MPCBB) . . . 175
5.4.5 TrustZone illegal access controller (TZIC) . . . 175
5.4.6 Power-on/reset state . . . 175

5.5 GTZC interrupts 176
5.6 GTZC1 TZSC registers 176
5.6.1 GTZC1 TZSC control register (GTZC1_TZSC_CR) 176
5.6.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1) 177
5.6.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2) 180
5.6.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3) 182
5.6.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1) 185
5.6.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2) 188
5.6.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3) 190
5.6.8 GTZC1 TZSC memory x subregion z watermark configuration register (GTZC1_TZSC_MPCWM1zCFGR) (z = A to B) 193
5.6.9 GTZC1 TZSC subregion A watermark register (GTZC1_TZSC_MPCWM1AR) 194
5.6.10 GTZC1 TZSC subregion B watermark register (GTZC1_TZSC_MPCWM1BR) 195
5.6.11 GTZC1 TZSC register map 195
5.7 GTZC1 TZIC registers 197
5.7.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1) 197
5.7.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2) 199
5.7.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3) 202
5.7.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4) 204
5.7.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1) 206
5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2) 210
5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3) 212
5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4) 215
5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1) 217
5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2) 219
5.7.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3) 221
5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4) 224
5.7.13 GTZC1 TZIC register map 226
5.8 GTZC1 MPCBBz registers (z = 1 to 3) 228
5.8.1 GTZC1 SRAMz MPCBB control register (GTZC1_MPCBBz_CR) (z = 1 to 3) 228

5.8.2 GTZC1 SRAMz MPCBB configuration lock register 1 (GTZC1_MPCBBz_CFGLOCK1) (z = 1 to 3) . . . 229
5.8.3 GTZC1 SRAMz MPCBB security configuration for super-block x register (GTZC1_MPCBBz_SECCFGRx) (z = 1 to 3) . . . 229
5.8.4 GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 3) . . . 230
5.8.5 GTZC1 MPCBBz register map (z = 1 to 3) . . . 230

6 RAMs configuration controller (RAMCFG) . . . 231
6.1 Introduction . . . 231
6.2 RAMCFG main features . . . 231
6.3 RAMCFG functional description . . . 231
6.3.1 Internal SRAMs features . . . 231
6.3.2 Error code correction (SRAM1, SRAM2, SRAM3) . . . 232
6.3.3 Write protection . . . 233
6.3.4 Software erase . . . 233
6.4 RAMCFG low-power modes . . . 233
6.5 RAMCFG interrupts . . . 234
6.6 RAMCFG registers . . . 234
6.6.1 RAMCFG memory x control register (RAMCFG_MxCR) . . . 234
6.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxIER) . . . 235
6.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR) . . . 235
6.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR) . . . 236
6.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR) . . . 236
6.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxICR) . . . 237
6.6.7 RAMCFG memory 2 write protection register 1 (RAMCFG_MxWPR1) . . . 237
6.6.8 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2) . . . 238
6.6.9 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR) . . . 238
6.6.10 RAMCFG memory x erase key register (RAMCFG_MxERKEYR) . . . 239
6.6.11 RAMCFG register map . . . 239

7 Embedded flash memory (FLASH) . . . 242
7.1 Introduction . . . 242
7.2 FLASH main features . . . 242

7.3 FLASH functional description 243
7.3.1 FLASH block diagram 243
7.3.2 FLASH signals 244
7.3.3 Flash memory architecture and usage 245
7.3.4 FLASH read operations 247
7.3.5 FLASH program operations 249
7.3.6 FLASH erase operations 252
7.3.7 PUF operations 256
7.3.8 FLASH parallel operations 258
7.3.9 FLASH error protections 258
7.3.10 OTP and RO memory access 258
7.3.11 Flash high-cycle data 261
7.3.12 Flash bank swapping 265
7.3.13 FLASH reset and clocks 270
7.4 FLASH option bytes 272
7.4.1 About option bytes 272
7.4.2 Option byte loading 272
7.4.3 Option-byte modification 272
7.4.4 Option-byte overview 274
7.4.5 Description of user and system option bytes 275
7.4.6 Description of data protection option bytes 276
7.4.7 Description of boot address option bytes 277
7.4.8 Specific rules for modifying option bytes 277
7.5 Option-byte key management 280
7.5.1 OBK loading 280
7.5.2 OBK access per HDPL level 280
7.5.3 OBK programming sequence 280
7.5.4 OBK programming finite state machine 283
7.5.5 OBK swap page 283
7.5.6 OBK alternate page erase 286
7.6 FLASH security and protections 286
7.6.1 TrustZone® security protection 287
7.6.2 Hide protection (HDP) 289
7.6.3 Block-based secure flash memory area protection 291
7.6.4 Block-based privileged flash memory area protection 292
7.6.5 Flash memory register privileged and unprivileged modes 293
7.6.6 Flash memory bank attributes in case of bank swap 294

7.6.7 Flash memory configuration protection 295
7.6.8 Write protection 296
7.6.9 Flash high-cycle data protections 296
7.6.10 Life cycle management 298
7.6.11 Product state transitions 299
7.6.12 OBK protection 301
7.6.13 One-time-programmable and read-only memory protections 302
7.7 System memory 303
7.7.1 Introduction 303
7.7.2 RSS user functions 303
7.8 FLASH low-power modes 310
7.9 FLASH error management 311
7.9.1 Introduction 311
7.9.2 Nonsecure write protection error (WRPERR) 312
7.9.3 Secure write protection error (WRPERR) 312
7.9.4 Nonsecure programming sequence error (PGSERR) 313
7.9.5 Secure programming sequence error (PGSERR) 315
7.9.6 Nonsecure strobe error (STRBERR) 316
7.9.7 Secure strobe error (STRBERR) 316
7.9.8 Nonsecure inconsistency error (INCERR) 316
7.9.9 Secure inconsistency error (INCERR) 317
7.9.10 Error correction code error (ECCC, ECCD) 318
7.9.11 Illegal access (ILAFM/ILAP) 319
7.9.12 Option-byte change error (OPTCHANGEERR) 319
7.9.13 Miscellaneous HardFault errors 320
7.9.14 OBK error cases (OBKERR, OBKWERR) 320
7.10 FLASH interrupts 321
7.11 FLASH registers 323
7.11.1 FLASH access control register (FLASH_ACR) 323
7.11.2 FLASH nonsecure key register (FLASH_KEYR) 324
7.11.3 FLASH secure key register (FLASH_SKEYR) 324
7.11.4 FLASH option key register (FLASH_OPTKEYR) 325
7.11.5 FLASH nonsecure OBK key register (FLASH_OBKKEYR) 325
7.11.6 FLASH secure OBK key register (FLASH_SOBKKEYR) 326
7.11.7 FLASH operation status register (FLASH_OPSR) 326
7.11.8 FLASH option control register (FLASH_OPTCR) 327

7.11.9 FLASH nonsecure status register (FLASH_SR) 328
7.11.10 FLASH secure status register (FLASH_SSR) 330
7.11.11 FLASH nonsecure control register (FLASH_CR) 332
7.11.12 FLASH secure control register (FLASH_SCR) 335
7.11.13 FLASH nonsecure clear control register (FLASH_CCR) 338
7.11.14 FLASH secure clear control register (FLASH_SCCR) 339
7.11.15 FLASH privilege configuration register (FLASH_PRIVCFGR) 340
7.11.16 FLASH nonsecure OBK configuration register (FLASH_OBKCFGR) 340
7.11.17 FLASH secure OBK configuration register (FLASH_SOBKCFGR) 342
7.11.18 FLASH HDP extension register (FLASH_HDPEXTR) 343
7.11.19 FLASH option status register (FLASH_OPTSR_CUR) 343
7.11.20 FLASH option status register (FLASH_OPTSR_PRG) 345
7.11.21 FLASH nonsecure EPOCH register (FLASH_EPOCHR_CUR) 346
7.11.22 FLASH nonsecure EPOCH register (FLASH_EPOCHR_PRG) 347
7.11.23 FLASH secure EPOCH register (FLASH_SEPOCHR_CUR) 347
7.11.24 FLASH secure EPOCH register (FLASH_SEPOCHR_PRG) 348
7.11.25 FLASH option status register 2 (FLASH_OPTSR2_CUR) 348
7.11.26 FLASH option status register 2 (FLASH_OPTSR2_PRG) 349
7.11.27 FLASH option status register 3 (FLASH_OPTSR3_CUR) 350
7.11.28 FLASH option status register 3 (FLASH_OPTSR3_PRG) 351
7.11.29 FLASH nonsecure boot register (FLASH_BOOTR_CUR) 352
7.11.30 FLASH nonsecure boot register (FLASH_BOOTR_PRG) 352
7.11.31 FLASH secure boot register (FLASH_SBOOTR_CUR) 353
7.11.32 FLASH secure boot register (FLASH_SBOOTR_PRG) 354
7.11.33 FLASH nonsecure OTP block lock (FLASH_OTPBLR_CUR) 354
7.11.34 FLASH nonsecure OTP block lock (FLASH_OTPBLR_PRG) 355
7.11.35 FLASH bootloader interface selection (FLASH_BL_COM_CFG_CUR) 356
7.11.36 FLASH bootloader interface selection (FLASH_BL_COM_CFG_PRG) 357
7.11.37 FLASH secure block based registers for Bank1 (FLASH_SECBB1R) 358
7.11.38 FLASH privilege block based registers for Bank1 (FLASH_PRIVBB1R) 358
7.11.39 FLASH security watermark for Bank1 (FLASH_SECWM1R_CUR) . . 359
7.11.40 FLASH security watermark for Bank1 (FLASH_SECWM1R_PRG) . . 359

7.11.41 FLASH write page group protection for Bank1 (FLASH_WRP1R_CUR) . . . 360
7.11.42 FLASH write page group protection for Bank1 (FLASH_WRP1R_PRG) . . . 361
7.11.43 FLASH data page configuration Bank1 (FLASH_EDATA1R_CUR) . . . 361
7.11.44 FLASH data page configuration Bank1 (FLASH_EDATA1R_PRG) . . . 362
7.11.45 FLASH HDP Bank1 configuration (FLASH_HDP1R_CUR) . . . 363
7.11.46 FLASH HDP Bank1 configuration (FLASH_HDP1R_PRG) . . . 363
7.11.47 FLASH ECC correction register (FLASH_ECCCORR) . . . 364
7.11.48 FLASH ECC detection register (FLASH_ECCDETR) . . . 365
7.11.49 FLASH ECC data (FLASH_ECCDR) . . . 366
7.11.50 FLASH secure block-based registers for Bank2 (FLASH_SECBB2R) . . . 366
7.11.51 FLASH privilege block-based registers for Bank2 (FLASH_PRIVBB2R) . . . 367
7.11.52 FLASH security watermark for Bank2 (FLASH_SECWM2R_CUR) . . 367
7.11.53 FLASH security watermark for Bank2 (FLASH_SECWM2R_PRG) . . 368
7.11.54 FLASH write page group protection for Bank2 (FLASH_WRP2R_CUR) . . . 368
7.11.55 FLASH write page group protection for Bank2 (FLASH_WRP2R_PRG) . . . 369
7.11.56 FLASH data page configuration Bank2 (FLASH_EDATA2R_CUR) . . . 370
7.11.57 FLASH data page configuration Bank2 (FLASH_EDATA2R_PRG) . . . 371
7.11.58 FLASH HDP Bank2 configuration (FLASH_HDP2R_CUR) . . . 371
7.11.59 FLASH HDP Bank2 configuration (FLASH_HDP2R_PRG) . . . 372
7.12 FLASH register map and reset values . . . 373

8 Instruction cache (ICACHE) . . . 378

8.1 ICACHE introduction . . . 378
8.2 ICACHE main features . . . 378
8.3 ICACHE implementation . . . 379
8.4 ICACHE functional description . . . 380
8.4.1 ICACHE block diagram . . . 380
8.4.2 ICACHE reset and clocks . . . 381
8.4.3 ICACHE TAG memory . . . 381
8.4.4 Direct-mapped ICACHE (1-way cache) . . . 382
8.4.5 ICACHE enable . . . 383
8.4.6 Cacheable and noncacheable traffic . . . 384

8.4.7 Address remapping . . . 385
8.4.8 Cacheable accesses . . . 387
8.4.9 Dual-master cache . . . 388
8.4.10 ICACHE security . . . 388
8.4.11 ICACHE maintenance . . . 388
8.4.12 ICACHE ECC support . . . 388
8.4.13 ICACHE performance monitoring . . . 390
8.4.14 ICACHE boot . . . 390
8.5 ICACHE low-power modes . . . 390
8.6 ICACHE error management and interrupts . . . 391
8.7 ICACHE registers . . . 392
8.7.1 ICACHE control register (ICACHE_CR) . . . 392
8.7.2 ICACHE status register (ICACHE_SR) . . . 393
8.7.3 ICACHE interrupt enable register (ICACHE_IER) . . . 394
8.7.4 ICACHE flag clear register (ICACHE_FCR) . . . 394
8.7.5 ICACHE hit monitor register (ICACHE_HMONR) . . . 395
8.7.6 ICACHE miss monitor register (ICACHE_MMONR) . . . 395
8.7.7 ICACHE region x configuration register (ICACHE_CRRx) . . . 396
8.7.8 ICACHE register map . . . 397

9 Data cache (DCACHE) . . . 398
9.1 DCACHE introduction . . . 398
9.2 DCACHE main features . . . 398
9.3 DCACHE implementation . . . 399
9.4 DCACHE functional description . . . 399
9.4.1 DCACHE block diagram . . . 400
9.4.2 DCACHE reset and clocks . . . 400
9.4.3 DCACHE TAG memory . . . 400
9.4.4 DCACHE enable . . . 402
9.4.5 Cacheable and noncacheable traffic . . . 402
9.4.6 Cacheable accesses . . . 403
9.4.7 DCACHE security . . . 405
9.4.8 DCACHE maintenance . . . 405
9.4.9 DCACHE ECC support . . . 407
9.4.10 DCACHE performance monitoring . . . 410
9.4.11 DCACHE boot . . . 411

9.5 DCACHE low-power modes 411
9.6 DCACHE error management and interrupts 411
9.7 DCACHE registers 412
9.7.1 DCACHE control register (DCACHE_CR) 412
9.7.2 DCACHE status register (DCACHE_SR) 414
9.7.3 DCACHE interrupt enable register (DCACHE_IER) 415
9.7.4 DCACHE flag clear register (DCACHE_FCR) 416
9.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR) 417
9.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR) 417
9.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR) 417
9.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR) 418
9.7.9 DCACHE command range start address register (DCACHE_CMDRSADDRR) 418
9.7.10 DCACHE command range end address register (DCACHE_CMDREADDRR) 418
9.7.11 DCACHE register map 419

10 Power control (PWR) 420
10.1 Introduction 420
10.2 PWR main features 420
10.3 PWR pins and internal signals 421
10.4 PWR power supplies and supply domains 422
10.4.1 External power supplies 424
10.4.2 Internal and external regulators 424
10.4.3 Power-up and power-down power sequences 426
10.4.4 Independent analog peripherals supply 427
10.4.5 USB transceivers supply 428
10.4.6 RTC domain 428
10.5 PWR system supply voltage regulation 428
10.5.1 SMPS and LDO embedded regulators 428
10.5.2 V_{CORE} supply versus reset, voltage scaling, and low-power modes . . 429
10.5.3 Embedded voltage regulator operating modes 429
10.5.4 Regulator bypass operating modes 429
10.6 PWR power supply and temperature supervision 430
10.6.1 Power-on reset (POR)/power-down reset (PDR) 430
10.6.2 Programmable voltage detector (PVD) 431
10.6.3 Analog voltage detector (AVD) 431

10.6.4 VDD voltage monitoring 432
10.6.5 Temperature monitoring 433
10.7 PWR management 434
10.7.1 Voltage scaling 434
10.7.2 Power management examples 435
10.8 Power modes 436
10.8.1 Slowing down system clocks 439
10.8.2 Peripheral clock gating 440
10.8.3 Low-power modes 440
10.8.4 Sleep mode 441
10.8.5 Stop mode 442
10.8.6 Standby mode 444
10.8.7 Power modes output pins 447
10.9 PWR security and privileged protection 448
10.9.1 PWR security protection 448
10.9.2 PWR privileged protection 449
10.10 PWR interrupts 450
10.11 PWR registers 451
10.11.1 PWR low power mode control register (PWR_LPMCR) 451
10.11.2 PWR status register (PWR_PMSR) 452
10.11.3 PWR voltage scaling control register (PWR_VOSCR) 453
10.11.4 PWR voltage scaling status register (PWR_VOSSR) 453
10.11.5 PWR tamper monitoring control register (PWR_TMCR) 454
10.11.6 PWR RTC domain control register (PWR_RTCCR) 455
10.11.7 PWR tamper monitoring status register (PWR_TMSR) 455
10.11.8 PWR USB Type-C® power delivery register (PWR_UCPDR) 456
10.11.9 PWR supply configuration control register (PWR_SCCR) 456
10.11.10 PWR voltage monitor control register (PWR_VMCR) 457
10.11.11 PWR USB supply control register (PWR_USBSCR) 458
10.11.12 PWR voltage monitor status register (PWR_VMSR) 459
10.11.13 PWR wake-up status clear register (PWR_WUSCR) 459
10.11.14 PWR wake-up status register (PWR_WUSR) 460
10.11.15 PWR wake-up configuration register (PWR_WUCR) 460
10.11.16 PWR I/O retention register (PWR_IORETR) 461
10.11.17 PWR security configuration register (PWR_SECCFGR) 462
10.11.18 PWR privilege configuration register (PWR_PRIVCFGR) 463

10.11.19 PWR register map 464

11 Reset and clock control (RCC) 466

11.1 RCC introduction 466

11.2 RCC main features 466

11.3 RCC functional description overview 467

11.3.1 RCC block diagram 467

11.3.2 RCC pins and internal signals 468

11.4 RCC reset functional description 470

11.4.1 Reset from/to external devices (NRST pin) 471

11.4.2 Application and system resets 471

11.4.3 RTC domain reset 472

11.4.4 CoreSight™ debug reset 472

11.4.5 Option bytes loading 472

11.4.6 Peripherals reset 472

11.4.7 Low-power mode security reset (lpwr_rst) 473

11.4.8 Reset coverage summary 474

11.4.9 Reset source identification 475

11.5 RCC clock functional description 475

11.5.1 Clock naming convention 478

11.5.2 Oscillators state summary 478

11.5.3 HSE oscillator 479

11.5.4 HSI oscillator 481

11.5.5 HSI48 oscillator 483

11.5.6 LSE oscillator 484

11.5.7 LSI oscillator 486

11.5.8 Clock security system (CSS) 487

11.5.9 Clock output generation (MCO1, MCO2) 490

11.5.10 PLL description 491

11.5.11 Intermediate clock generation (ICLKx) 494

11.5.12 System clock (**sys_ck**) 496

11.5.13 Handling clock generators in Stop and Standby modes 498

11.5.14 Peripheral clock distribution 499

11.5.15 Clock frequency measurement using TIMx 531

11.5.16 Peripheral clock gating 533

11.5.17 Bus and CPU clocks 535

11.6 RCC security and privilege functional description 536
11.6.1 TrustZone® security protection modes 536
11.6.2 Privileged protection modes 539
11.6.3 Clock protection 540
11.7 RCC interrupts 541
11.8 RCC registers 545
11.8.1 RCC clock control register (RCC_CR) 545
11.8.2 RCC HSI calibration register (RCC_HSICFGR) 548
11.8.3 RCC clock recovery RC register (RCC_CRRCR) 549
11.8.4 RCC clock configuration register1 (RCC_CFGR1) 550
11.8.5 RCC CPU domain clock configuration register 2 (RCC_CFGR2) 553
11.8.6 RCC PLL clock source selection register (RCC_PLL1CFGR) 556
11.8.7 RCC PLL clock source selection register (RCC_PLL2CFGR) 558
11.8.8 RCC PLL1 divider register (RCC_PLL1DIVR) 560
11.8.9 RCC PLL1 fractional divider register (RCC_PLL1FRACR) 562
11.8.10 RCC PLL2 divider register (RCC_PLL2DIVR) 563
11.8.11 RCC PLL2 fractional divider register (RCC_PLL2FRACR) 565
11.8.12 RCC interrupt enable register (RCC_IER) 566
11.8.13 RCC interrupt flag register (RCC_IFR) 568
11.8.14 RCC Interrupt clear register (RCC_ICR) 570
11.8.15 RCC AHB1 reset register (RCC_AHB1RSTR) 572
11.8.16 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) 574
11.8.17 RCC AHB4 peripheral reset register (RCC_AHB4RSTR) 576
11.8.18 RCC APB1 peripheral low reset register (RCC_APB1LRSTR) 579
11.8.19 RCC APB1 peripheral high reset register (RCC_APB1HRSTR) 582
11.8.20 RCC APB2 peripheral reset register (RCC_APB2RSTR) 583
11.8.21 RCC APB3 peripheral reset register (RCC_APB3RSTR) 585
11.8.22 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR) 588
11.8.23 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR) 590
11.8.24 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR) 592
11.8.25 RCC AHB4 peripheral clock enable register (RCC_AHB4ENR) 593
11.8.26 RCC APB1 peripheral clock enable register (RCC_APB1LENR) 596
11.8.27 RCC APB1 peripheral clock enable register (RCC_APB1HENR) 599
11.8.28 RCC APB2 peripheral clock enable register (RCC_APB2ENR) 600
11.8.29 RCC APB3 peripheral clock enable register (RCC_APB3ENR) 602
11.8.30 RCC AHB1 peripheral low-power clock enable register (RCC_AHB1LPENR) 606

11.8.31 RCC AHB2 peripheral low-power clock enable register (RCC_AHB2LPENR) 608
11.8.32 RCC AHB3 peripheral low-power clock enable register (RCC_AHB3LPENR) 610
11.8.33 RCC AHB4 peripheral low-power clock enable register (RCC_AHB4LPENR) 611
11.8.34 RCC APB1 peripheral low-power clock enable register (RCC_APB1LLPENR) 614
11.8.35 RCC APB1 peripheral low-power clock enable register (RCC_APB1HLPENR) 617
11.8.36 RCC APB2 peripheral low-power clock enable register (RCC_APB2LPENR) 618
11.8.37 RCC APB3 peripheral low-power clock enable register (RCC_APB3LPENR) 620
11.8.38 RCC kernel clock configuration register (RCC_CCIPR1) 624
11.8.39 RCC kernel clock configuration register (RCC_CCIPR2) 627
11.8.40 RCC kernel clock configuration register (RCC_CCIPR3) 630
11.8.41 RCC kernel clock configuration register (RCC_CCIPR4) 633
11.8.42 RCC kernel clock configuration register (RCC_CCIPR5) 636
11.8.43 RCC RTC domain control register (RCC_RTCCR) 638
11.8.44 RCC reset status register (RCC_RSR) 641
11.8.45 RCC secure configuration register (RCC_SECCFGR) 643
11.8.46 RCC privilege configuration register (RCC_PRIVCFGR) 645
11.8.47 RCC register map 645

12 Clock recovery system (CRS) 652

12.1 CRS introduction 652
12.2 CRS main features 652
12.3 CRS implementation 652
12.4 CRS functional description 653
12.4.1 CRS block diagram 653
12.4.2 CRS internal signals 653
12.4.3 Synchronization input 654
12.4.4 Frequency error measurement 654
12.4.5 Frequency error evaluation and automatic trimming 655
12.4.6 CRS initialization and configuration 656
12.5 CRS in low-power modes 657
12.6 CRS interrupts 657

12.7 CRS registers . . . 657
12.7.1 CRS control register (CRS_CR) . . . 657
12.7.2 CRS configuration register (CRS_CFGR) . . . 659
12.7.3 CRS interrupt and status register (CRS_ISR) . . . 660
12.7.4 CRS interrupt flag clear register (CRS_ICR) . . . 662
12.7.5 CRS register map . . . 662

13 General-purpose I/Os (GPIO) . . . 664
13.1 Introduction . . . 664
13.2 GPIO main features . . . 664
13.3 GPIO functional description . . . 664
13.3.1 General-purpose I/O (GPIO) . . . 666
13.3.2 I/O pin alternate function multiplexer and mapping . . . 667
13.3.3 I/O port control registers . . . 667
13.3.4 I/O port data registers . . . 668
13.3.5 I/O data bitwise handling . . . 668
13.3.6 GPIO locking mechanism . . . 668
13.3.7 I/O alternate function input/output . . . 668
13.3.8 External interrupt/wake-up lines . . . 669
13.3.9 Input configuration . . . 669
13.3.10 Output configuration . . . 670
13.3.11 Alternate function configuration . . . 670
13.3.12 Analog configuration . . . 671
13.3.13 Using the HSE or LSE oscillator pins as GPIOs . . . 671
13.3.14 Using the GPIO pins in the RTC domain . . . 672
13.3.15 Privileged and unprivileged modes . . . 672
13.3.16 SMBUS 3.3 compliant mode for I2C pins . . . 672
13.3.17 I/Os state retention during standby mode . . . 672
13.3.18 TrustZone security . . . 672
13.3.19 I/O compensation cell . . . 674
13.4 GPIO registers . . . 675
13.4.1 GPIO port mode register (GPIOx_MODER) (x = A to H) . . . 675
13.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to H) . . . 675
13.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to H) . . . 676
13.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to H) . . . 676
13.4.5 GPIO port input data register (GPIOx_IDR) (x = A to H) . . . 677
13.4.6 GPIO port output data register (GPIOx_ODR) (x = A to H) . . . 677

13.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to H) 678
13.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to H) . . . 678
13.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to H) 679
13.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to H) . . . 680
13.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to H) 681
13.4.12 GPIO secure configuration register (GPIOx_SECCFGR) (x = A to H) . 682
13.4.13 GPIO register map 683

14 System configuration, boot, and security (SBS) 685

14.1 SBS introduction 685
14.2 SBS main features 685
14.3 SBS functional description 686
14.3.1 SBS block diagram 686
14.3.2 SBS signals 687
14.3.3 SBS reset and clocks 687
14.3.4 SBS system configuration 688
14.3.5 SBS SMBUS 3.3 control 689
14.3.6 SBS boot control 689
14.3.7 SBS debug control 692
14.3.8 SBS hardware secure storage control 694
14.3.9 SBS flash banks control 696
14.4 SBS interrupts 696
14.5 SBS registers 697
14.5.1 SBS temporal isolation control register (SBS_HDPLCR) 697
14.5.2 SBS temporal isolation status register (SBS_HDPLSR) 697
14.5.3 SBS next HDPL control register (SBS_NEXTHDPLCR) 698
14.5.4 SBS debug control register (SBS_DBGCR) 698
14.5.5 SBS debug lock register (SBS_DBGLOCKR) 699
14.5.6 SBS debug control register 2 (SBS_DBGCR2) 699
14.5.7 SBS RSS command register (SBS_RSSCMDR) 700
14.5.8 SBS EPOCH selection control register (SBS_EPOCHSELCR) 700
14.5.9 SBS security mode configuration control register (SBS_SECCFGR) . 701
14.5.10 SBS product mode and configuration register (SBS_PMCR) 702
14.5.11 SBS FPU interrupt mask register (SBS_FPUIMR) 703
14.5.12 SBS memory erase status register (SBS_MESR) 703
14.5.13 SBS compensation cell for I/Os control and status register (SBS_CCCSR) 704

14.5.14 SBS compensation cell for I/Os value register (SBS_CCVALR) 705
14.5.15 SBS compensation cell for I/Os software code register (SBS_CCSWCR) 705
14.5.16 SBS Class B register (SBS_CFGR2) 706
14.5.17 SBS CPU nonsecure lock register (SBS_CNSLCKR) 707
14.5.18 SBS CPU secure lock register (SBS_CSLCKR) 708
14.5.19 SBS ECC NMI mask register (SBS_ECCNMIR) 709
14.5.20 SBS debug control register 2 (SBS_SWAPLOCKR) 709
14.5.21 SBS debug control register 2 (SBS_FBMODER) 710
14.5.22 SBS register map 710

15 Peripherals interconnect matrix 713

15.1 Interconnect matrix introduction 713
15.2 Connection summary 713
15.3 Interconnection details 718
15.3.1 Master to slave interconnection for timers 718
15.3.2 Triggers to ADCs 718
15.3.3 ADC analog watchdogs as triggers to timers 719
15.3.4 From comparators to timers (TIMx, HRTIM, LPTIM) 720
15.3.5 Triggers to DAC 721
15.3.6 Clock sources to timers 722
15.3.7 Triggers to low-power timers 723
15.3.8 RTC wake-up as inputs to timers 723
15.3.9 System errors as break signals to timers and HRTIM 723
15.3.10 From timers to comparators 724
15.3.11 Triggers to GPDMA1/2 724
15.3.12 Internal analog signals to ADCs 725
15.3.13 Internal analog signals to COMPs 725
15.3.14 Internal analog signals to OPAMPs 725
15.3.15 Clock source for the DAC sample and hold mode 726
15.3.16 Internal tamper sources 726
15.3.17 Output from tamper to RTC 726
15.3.18 PLAY inputs/outputs from/to internal peripherals 726
15.3.19 Encryption keys to AES/SAES 727
15.3.20 From USB to timers 727
15.3.21 CRS clock sources 727
15.3.22 Internal sources for EXTI lines 728

15.3.23 Internal signals to POSENC 728

16 General purpose direct memory access controller (GPDMA) 729

16.1 GPDMA introduction 729

16.2 GPDMA main features 729

16.3 GPDMA implementation 730

16.3.1 GPDMA instances 730

16.3.2 GPDMA channels 730

16.3.3 GPDMA in low-power modes 731

16.3.4 GPDMA requests 731

16.3.5 GPDMA block requests 736

16.3.6 GPDMA channels with peripheral early termination 736

16.3.7 GPDMA triggers 737

16.4 GPDMA functional description 739

16.4.1 GPDMA block diagram 739

16.4.2 GPDMA channel state and direct programming without any linked-list 739

16.4.3 GPDMA channel suspend and resume 740

16.4.4 GPDMA channel abort and restart 741

16.4.5 GPDMA channel terminate and drain FIFO 742

16.4.6 GPDMA linked-list data structure 743

16.4.7 Linked-list item transfer execution 747

16.4.8 GPDMA channel state and linked-list programming in run-to-completion mode 747

16.4.9 GPDMA channel state and linked-list programming in link step mode 751

16.4.10 GPDMA channel state and linked-list programming 758

16.4.11 GPDMA FIFO-based transfers 760

16.4.12 GPDMA transfer request and arbitration 767

16.4.13 GPDMA triggered transfer 771

16.4.14 GPDMA circular buffering with linked-list programming 772

16.4.15 GPDMA transfer in peripheral flow-control mode 774

16.4.16 GPDMA byte-based padding at the block level 775

16.4.17 GPDMA secure/nonsecure channel 776

16.4.18 GPDMA privileged/unprivileged channel 777

16.4.19 GPDMA error management 778

16.5 GPDMA in debug mode 779

16.6 GPDMA in low-power modes 779
16.7 GPDMA interrupts 780
16.8 GPDMA registers 781
16.8.1 GPDMA secure configuration register (GPDMA_SECCFGR) 781
16.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR) 782
16.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR) 782
16.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR) 783
16.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR) . . 784
16.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR) 784
16.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR) 785
16.8.8 GPDMA channel x status register (GPDMA_CxSR) 786
16.8.9 GPDMA channel x control register (GPDMA_CxCR) 788
16.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1) 790
16.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2) 794
16.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1) 798
16.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1) 799
16.8.14 GPDMA channel x source address register (GPDMA_CxSAR) 801
16.8.15 GPDMA channel x destination address register (GPDMA_CxDAR) . . 803
16.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3) 804
16.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2) 805
16.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR) 806
16.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR) 808
16.8.20 GPDMA register map 809

17 General purpose direct memory access controller (GPDMA) 812

17.1 GPDMA introduction 812
17.2 GPDMA main features 812
17.3 GPDMA implementation 813
17.3.1 GPDMA instances 813
17.3.2 GPDMA channels 813
17.3.3 GPDMA in low-power modes 814
17.3.4 GPDMA requests 814
17.3.5 GPDMA block requests 819
17.3.6 GPDMA channels with peripheral early termination 819

17.3.7 GPDMA triggers . . . 820
17.4 GPDMA functional description . . . 822
17.4.1 GPDMA block diagram . . . 822
17.4.2 GPDMA channel state and direct programming without any linked-list . . . 822
17.4.3 GPDMA channel suspend and resume . . . 823
17.4.4 GPDMA channel abort and restart . . . 824
17.4.5 GPDMA channel terminate and drain FIFO . . . 825
17.4.6 GPDMA linked-list data structure . . . 826
17.4.7 Linked-list item transfer execution . . . 830
17.4.8 GPDMA channel state and linked-list programming in run-to-completion mode . . . 830
17.4.9 GPDMA channel state and linked-list programming in link step mode . . . 834
17.4.10 GPDMA channel state and linked-list programming . . . 841
17.4.11 GPDMA FIFO-based transfers . . . 843
17.4.12 GPDMA transfer request and arbitration . . . 850
17.4.13 GPDMA triggered transfer . . . 854
17.4.14 GPDMA circular buffering with linked-list programming . . . 855
17.4.15 GPDMA transfer in peripheral flow-control mode . . . 857
17.4.16 GPDMA byte-based padding at the block level . . . 858
17.4.17 GPDMA secure/nonsecure channel . . . 859
17.4.18 GPDMA privileged/unprivileged channel . . . 860
17.4.19 GPDMA error management . . . 861
17.5 GPDMA in debug mode . . . 862
17.6 GPDMA in low-power modes . . . 862
17.7 GPDMA interrupts . . . 863
17.8 GPDMA registers . . . 864
17.8.1 GPDMA secure configuration register (GPDMA_SECCFGR) . . . 864
17.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR) . . . 865
17.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR) . . . 865
17.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR) . . . 866
17.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR) . . . 867
17.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR) . . . 867
17.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR) . . . 868
17.8.8 GPDMA channel x status register (GPDMA_CxSR) . . . 869

17.8.9 GPDMA channel x control register (GPDMA_CxCR) 871
17.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1) 873
17.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2) 877
17.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1) 881
17.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1) 882
17.8.14 GPDMA channel x source address register (GPDMA_CxSAR) 884
17.8.15 GPDMA channel x destination address register (GPDMA_CxDAR) . . 886
17.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3) 887
17.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2) 888
17.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR) 889
17.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR) 891
17.8.20 GPDMA register map 892

18 Nested vectored interrupt controller (NVIC) 895
18.1 NVIC main features 895
18.2 SysTick calibration value register 895
18.3 Interrupt and exception vectors 895

19 Extended interrupts and event controller (EXTI) 903
19.1 EXTI main features 903
19.2 EXTI block diagram 903
19.2.1 EXTI connections between peripherals and CPU 905
19.2.2 EXTI interrupt/event mapping 905
19.3 EXTI functional description 907
19.3.1 EXTI configurable event input wake-up 907
19.3.2 EXTI direct event input wake-up 908
19.3.3 EXTI mux selection 908
19.4 EXTI functional behavior 909
19.5 EXTI event protection 910
19.5.1 EXTI security protection 910
19.5.2 EXTI privilege protection 911
19.6 EXTI registers 911
19.6.1 EXTI rising trigger selection register (EXTI_RTSR1) 911
19.6.2 EXTI falling trigger selection register (EXTI_FTSR1) 912
19.6.3 EXTI software interrupt event register (EXTI_SWIER1) 913

19.6.4 EXTI rising edge pending register (EXTI_RPR1) 913
19.6.5 EXTI falling edge pending register (EXTI_FPR1) 914
19.6.6 EXTI security configuration register (EXTI_SECCFGR1) 915
19.6.7 EXTI privilege configuration register (EXTI_PRIVCFGR1) 915
19.6.8 EXTI rising trigger selection register 2 (EXTI_RTSR2) 916
19.6.9 EXTI falling trigger selection register 2 (EXTI_FTSR2) 917
19.6.10 EXTI software interrupt event register 2 (EXTI_SWIER2) 918
19.6.11 EXTI rising edge pending register 2 (EXTI_RPR2) 920
19.6.12 EXTI falling edge pending register 2 (EXTI_FPR2) 921
19.6.13 EXTI security configuration register 2 (EXTI_SECCFGR2) 922
19.6.14 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2) 923
19.6.15 EXTI external interrupt selection register (EXTI_EXTICR1) 924
19.6.16 EXTI external interrupt selection register (EXTI_EXTICR2) 926
19.6.17 EXTI external interrupt selection register (EXTI_EXTICR3) 928
19.6.18 EXTI external interrupt selection register (EXTI_EXTICR4) 929
19.6.19 EXTI lock register (EXTI_LOCKR) 931
19.6.20 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1) 932
19.6.21 EXTI CPU wake-up with event mask register (EXTI_EMR1) 932
19.6.22 EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2) 933
19.6.23 EXTI CPU wake-up with event mask register 2 (EXTI_EMR2) 934
19.6.24 EXTI register map 936

20 Cyclic redundancy check calculation unit (CRC) 938

20.1 CRC introduction 938
20.2 CRC main features 938
20.3 CRC functional description 939
20.3.1 CRC block diagram 939
20.3.2 CRC internal signals 939
20.3.3 CRC operation 939
20.4 CRC registers 941
20.4.1 CRC data register (CRC_DR) 941
20.4.2 CRC independent data register (CRC_IDR) 941
20.4.3 CRC control register (CRC_CR) 942
20.4.4 CRC initial value (CRC_INIT) 943
20.4.5 CRC polynomial (CRC_POL) 943
20.4.6 CRC register map 944

21 CORDIC coprocessor (CORDIC) 945
21.1 CORDIC introduction 945
21.2 CORDIC main features 945
21.3 CORDIC functional description 945
21.3.1 General description 945
21.3.2 CORDIC functions 945
21.3.3 Fixed point representation 952
21.3.4 Scaling factor 952
21.3.5 Precision 953
21.3.6 Zero-overhead mode 956
21.3.7 Polling mode 957
21.3.8 Interrupt mode 958
21.3.9 DMA mode 958
21.4 CORDIC registers 959
21.4.1 CORDIC control/status register (CORDIC_CSR) 959
21.4.2 CORDIC argument register (CORDIC_WDATA) 961
21.4.3 CORDIC result register (CORDIC_RDATA) 962
21.4.4 CORDIC register map 962

22 Filter math accelerator (FMAC) 963
22.1 FMAC introduction 963
22.2 FMAC main features 963
22.3 FMAC functional description 964
22.3.1 General description 964
22.3.2 Local memory and buffers 965
22.3.3 Input buffers 965
22.3.4 Output buffer 968
22.3.5 Initialization functions 970
22.3.6 Filter functions 971
22.3.7 Fixed point representation 975
22.3.8 Implementing FIR filters with the FMAC 975
22.3.9 Implementing IIR filters with the FMAC 977
22.3.10 Examples of filter initialization 979
22.3.11 Examples of filter operation 980
22.3.12 Filter design tips 982
22.4 FMAC registers 983

22.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG) 983
22.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG) 983
22.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG) 984
22.4.4 FMAC parameter register (FMAC_PARAM) 985
22.4.5 FMAC control register (FMAC_CR) 986
22.4.6 FMAC status register (FMAC_SR) 987
22.4.7 FMAC write data register (FMAC_WDATA) 988
22.4.8 FMAC read data register (FMAC_RDATA) 989
22.4.9 FMAC register map 989

23 Programmable logic array (PLAY) 991

23.1 PLAY introduction 991
23.2 PLAY main features 991
23.3 PLAY functional description 992
23.3.1 PLAY block diagram 992
23.3.2 PLAY pins and internal signals 992
23.3.3 PLAY reset and clocks 998
23.3.4 Input selection 999
23.3.5 Input filtering and edge detection 999
23.3.6 Software programmable inputs 1002
23.3.7 Logic elements 1002
23.3.8 Input interconnect 1006
23.3.9 Output interconnect 1011
23.3.10 Flags 1012
23.3.11 Miscellaneous status bits 1013
23.4 PLAY in low-power modes 1013
23.5 PLAY interrupts 1013
23.6 PLAY registers 1014
23.6.1 PLAY configuration control register (PLAY_CFGCR) 1015
23.6.2 PLAY security attributes configuration register (PLAY_SECCFGR) . 1015
23.6.3 PLAY privilege attributes configuration register (PLAY_PRIVCFGR) 1016
23.6.4 PLAY filter x configuration register (PLAY_FILTxCFG) 1017
23.6.5 PLAY logic element x configuration register 1 (PLAY_LExCFG1) . . . 1018
23.6.6 PLAY logic element x configuration register 2 (PLAY_LExCFG2) . . . 1018
23.6.7 PLAY output x configuration register (PLAY_OUTxCFG) 1019
23.6.8 PLAY software input status register (PLAY_SWIN) 1020

23.6.9 PLAY software input set register (PLAY_SWINSET) 1020
23.6.10 PLAY software input clear register (PLAY_SWINCLR) 1021
23.6.11 PLAY output status register (PLAY_OSR) 1022
23.6.12 PLAY flag status register (PLAY_FLSTAT) 1022
23.6.13 PLAY flag set register (PLAY_FLSET) 1023
23.6.14 PLAY flag clear register (PLAY_FLCLR) 1024
23.6.15 PLAY flag interrupt enable register (PLAY_FLIER) 1024
23.6.16 PLAY flag control register (PLAY_FLCTL) 1025
23.6.17 PLAY miscellaneous status register (PLAY_MSR) 1026
23.6.18 PLAY interrupt status register (PLAY_ISR) 1026
23.6.19 PLAY interrupt clear register (PLAY_ICR) 1027
23.6.20 PLAY interrupt enable register (PLAY_IER) 1028
23.7 PLAY register map 1028

24 Extended-SPI interface (XSPI) 1031
24.1 XSPI introduction 1031
24.2 XSPI main features 1031
24.3 XSPI implementation 1032
24.4 XSPI functional description 1033
24.4.1 XSPI block diagram 1033
24.4.2 XSPI pins and internal signals 1034
24.4.3 XSPI interface to memory modes 1035
24.4.4 XSPI regular-command protocol 1035
24.4.5 XSPI regular-command protocol signal interface 1038
24.4.6 HyperBus protocol 1042
24.4.7 Specific features 1045
24.4.8 XSPI operating mode introduction 1046
24.4.9 XSPI indirect mode 1046
24.4.10 XSPI automatic status-polling mode 1048
24.4.11 XSPI memory-mapped mode 1049
24.4.12 XSPI configuration introduction 1050
24.4.13 XSPI system configuration 1050
24.4.14 XSPI device configuration 1051
24.4.15 XSPI regular-command mode configuration 1054
24.4.16 XSPI HyperBus protocol configuration 1056
24.4.17 XSPI error management 1057
24.4.18 XSPI BUSY and ABORT 1057

24.4.19 XSPI reconfiguration or deactivation 1058
24.4.20 NCS behavior 1058
24.4.21 Software control of two external memories 1058
24.4.22 Hardware-controlled extended memory support 1059
24.5 Address alignment and data number 1060
24.6 XSPI interrupts 1061
24.7 XSPI registers 1062
24.7.1 XSPI control register (XSPI_CR) 1062
24.7.2 XSPI device configuration register 1 (XSPI_DCR1) 1065
24.7.3 XSPI device configuration register 2 (XSPI_DCR2) 1067
24.7.4 XSPI device configuration register 3 (XSPI_DCR3) 1068
24.7.5 XSPI device configuration register 4 (XSPI_DCR4) 1068
24.7.6 XSPI status register (XSPI_SR) 1069
24.7.7 XSPI flag clear register (XSPI_FCR) 1070
24.7.8 XSPI data length register (XSPI_DLR) 1070
24.7.9 XSPI address register (XSPI_AR) 1071
24.7.10 XSPI data register (XSPI_DR) 1072
24.7.11 XSPI polling status mask register (XSPI_PSMKR) 1072
24.7.12 XSPI polling status match register (XSPI_PSMAR) 1073
24.7.13 XSPI polling interval register (XSPI_PIR) 1073
24.7.14 XSPI communication configuration register (XSPI_CCR) 1073
24.7.15 XSPI timing configuration register (XSPI_TCR) 1075
24.7.16 XSPI instruction register (XSPI_IR) 1076
24.7.17 XSPI alternate bytes register (XSPI_ABR) 1076
24.7.18 XSPI low-power timeout register (XSPI_LPTR) 1077
24.7.19 XSPI wrap communication configuration register (XSPI_WPCCR) 1077
24.7.20 XSPI wrap timing configuration register (XSPI_WPTCR) 1079
24.7.21 XSPI wrap instruction register (XSPI_WPIR) 1080
24.7.22 XSPI wrap alternate bytes register (XSPI_WPABR) 1080
24.7.23 XSPI write communication configuration register (XSPI_WCCR) 1081
24.7.24 XSPI write timing configuration register (XSPI_WTCR) 1083
24.7.25 XSPI write instruction register (XSPI_WIR) 1083
24.7.26 XSPI write alternate bytes register (XSPI_WABR) 1084
24.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR) 1084
24.7.28 XSPI register map 1085

25 Delay block (DLYB) 1088
25.1 DLYB introduction 1088
25.2 DLYB main features 1088
25.3 DLYB implementation 1088
25.4 DLYB functional description 1088
25.4.1 DLYB diagram 1088
25.4.2 DLYB pins and internal signals 1089
25.4.3 General description 1089
25.4.4 Delay line length configuration procedure 1090
25.4.5 Output clock phase configuration procedure 1091
25.5 DLYB registers 1092
25.5.1 DLYB control register (DLYB_CR) 1092
25.5.2 DLYB configuration register (DLYB_CFGR) 1092
25.5.3 DLYB register map 1093

26 Analog-to-digital converters (ADC) 1094
26.1 ADC introduction 1094
26.2 ADC main features 1094
26.3 ADC implementation 1096
26.4 ADC functional description 1097
26.4.1 ADC block diagram 1097
26.4.2 ADC pins and internal signals 1098
26.4.3 ADC clocks 1103
26.4.4 ADC connectivity 1105
26.4.5 Slave AHB interface 1108
26.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) 1108
26.4.7 Single-ended and differential input channels 1108
26.4.8 Pseudodifferential mode 1109
26.4.9 Calibration (ADCAL, ADCALDIF, ADC_CALFACT) 1109
26.4.10 ADC on-off control (ADEN, ADDIS, ADRDY) 1112
26.4.11 Constraints when writing the ADC control bits 1113
26.4.12 Channel selection (ADC_SQRy, ADC_JSQR) 1114
26.4.13 Channel preselection register (ADC_PCSEL) 1115
26.4.14 Channel-wise programmable sampling time (SMPR1, SMPR2) 1115
26.4.15 Single conversion mode 1117

26.4.16 Continuous conversion mode (CONT = 1) 1118
26.4.17 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) 1118
26.4.18 Starting conversions (ADSTART, JADSTART) 1120
26.4.19 Timing 1121
26.4.20 Stopping an ongoing conversion (ADSTP, JADSTP) 1122
26.4.21 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) 1123
26.4.22 Injected channel management 1125
26.4.23 Queue of context for injected conversions 1126
26.4.24 Programmable resolution (RES) - fast conversion mode 1134
26.4.25 End of conversion and end of sampling phase (EOC, JEOC, EOSMP) 1135
26.4.26 End of conversion sequence (EOS, JEOS) 1135
26.4.27 Timing diagram examples (single/continuous modes, hardware/software triggers) 1136
26.4.28 Data management 1138
26.4.29 Dynamic low-power features 1146
26.4.30 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT) 1150
26.4.31 Oversampler 1155
26.4.32 Parallel injected oversampling mode 1161
26.4.33 Dual ADC modes 1163
26.4.34 Temperature sensor 1178
26.4.35 Monitoring the internal voltage reference 1180
26.4.36 Monitoring the OPAMP output voltage 1182
26.4.37 Monitoring the supply voltage 1182
26.5 ADC interrupts 1182
26.6 ADC registers (for each ADC) 1183
26.6.1 ADC interrupt and status register (ADC_ISR) 1183
26.6.2 ADC interrupt enable register (ADC_IER) 1186
26.6.3 ADC control register (ADC_CR) 1188
26.6.4 ADC configuration register (ADC_CFGR1) 1191
26.6.5 ADC configuration register 2 (ADC_CFGR2) 1194
26.6.6 ADC sample time register 1 (ADC_SMPR1) 1197
26.6.7 ADC sample time register 2 (ADC_SMPR2) 1198
26.6.8 ADC channel preselection register (ADC_PCSEL) 1198
26.6.9 ADC regular sequence register 1 (ADC_SQR1) 1199

26.6.10 ADC regular sequence register 2 (ADC_SQR2) 1200
26.6.11 ADC regular sequence register 3 (ADC_SQR3) 1201
26.6.12 ADC regular sequence register 4 (ADC_SQR4) 1202
26.6.13 ADC regular data register (ADC_DR) 1203
26.6.14 ADC injected sequence register (ADC_JSQR) 1203
26.6.15 ADC offset y configuration register (ADC_OFCFGRy) 1205
26.6.16 ADC offset y register (ADC_OFRy) 1206
26.6.17 ADC gain compensation register (ADC_GCOMP) 1206
26.6.18 ADC analog watchdog y source channel register (ADC_AWDySR) . 1207
26.6.19 ADC injected channel y data register (ADC_JDRy) 1208
26.6.20 ADC configuration register 3 (ADC_CFGR3) 1208
26.6.21 ADC injected oversampling counter register 1 (ADC_JOVSCR1) . . . 1210
26.6.22 ADC injected oversampling counter register 2 (ADC_JOVSCR2) . . . 1211
26.6.23 ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . 1212
26.6.24 ADC analog watchdog 3 configuration register (ADC_AWD3CR) . . . 1212
26.6.25 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR) 1213
26.6.26 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR) 1213
26.6.27 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR) 1214
26.6.28 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR) 1214
26.6.29 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR) 1215
26.6.30 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR) 1215
26.6.31 ADC differential mode selection register (ADC_DIFSEL) 1216
26.6.32 ADC calibration factors (ADC_CALFACT) 1216
26.6.33 ADC option register (ADC_OR) 1218
26.7 ADC common registers 1219
26.7.1 ADC common status register (ADCC_CSR) 1219
26.7.2 ADC common control register (ADCC_CCR) 1221
26.7.3 ADC common regular data register for dual mode (ADCC_CDR) 1222
26.7.4 ADC common regular data register for dual mode (ADCC_CDR2) 1223
26.8 ADC register map 1223

27 Digital temperature sensor (DTS) 1228
27.1 DTS introduction 1228

27.2 DTS main features 1228
27.3 DTS functional description 1229
27.3.1 DTS block diagram 1229
27.3.2 DTS internal signals 1229
27.3.3 DTS block operation 1230
27.3.4 Operating modes 1230
27.3.5 Calibration 1230
27.3.6 Prescaler 1230
27.3.7 Temperature measurement principles 1231
27.3.8 Sampling time 1232
27.3.9 Quick measurement mode 1232
27.3.10 Trigger input 1233
27.3.11 On-off control and ready flag 1233
27.3.12 Temperature measurement sequence 1234
27.4 DTS low-power modes 1235
27.5 DTS interrupts 1235
27.5.1 Temperature window comparator 1235
27.5.2 Synchronous interrupt 1235
27.5.3 Asynchronous wake-up 1235
27.6 DTS registers 1236
27.6.1 Temperature sensor configuration register 1 (DTS_CFGR1) 1236
27.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1) 1237
27.6.3 Temperature sensor ramp value register (DTS_RAMPVALR) 1238
27.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1) 1238
27.6.5 Temperature sensor data register (DTS_DR) 1239
27.6.6 Temperature sensor status register (DTS_SR) 1239
27.6.7 Temperature sensor interrupt enable register (DTS_ITENR) 1240
27.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR) 1242
27.6.9 Temperature sensor option register (DTS_OR) 1242
27.6.10 DTS register map 1243

28 Digital-to-analog converter (DAC) 1245
28.1 DAC introduction 1245
28.2 DAC main features 1245
28.3 DAC implementation 1246
28.4 DAC functional description 1247

28.4.1 DAC block diagram . . . 1247
28.4.2 DAC pins and internal signals . . . 1247
28.4.3 DAC channel enable . . . 1251
28.4.4 DAC data format . . . 1251
28.4.5 DAC conversion . . . 1252
28.4.6 DAC output voltage . . . 1253
28.4.7 DAC trigger selection . . . 1253
28.4.8 DMA requests . . . 1254
28.4.9 Noise generation . . . 1254
28.4.10 Triangle-wave generation . . . 1256
28.4.11 DAC sawtooth wave generation . . . 1257
28.4.12 DAC extended sawtooth wave generation . . . 1258
28.4.13 DAC channel modes . . . 1261
28.4.14 DAC channel buffer calibration . . . 1265
28.4.15 DAC channel conversion modes . . . 1266
28.5 DAC in low-power modes . . . 1267
28.6 DAC interrupts . . . 1268
28.7 DAC registers . . . 1268
28.7.1 DAC control register (DAC_CR) . . . 1268
28.7.2 DAC software trigger register (DAC_SWTRGR) . . . 1270
28.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) . . . 1270
28.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) . . . 1271
28.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) . . . 1271
28.7.6 DAC channel1 data output register (DAC_DOR1) . . . 1272
28.7.7 DAC status register (DAC_SR) . . . 1272
28.7.8 DAC calibration control register (DAC_CCR) . . . 1273
28.7.9 DAC mode control register (DAC_MCR) . . . 1273
28.7.10 DAC channel1 sample and hold sample time register (DAC_SHSR1) . . . 1274
28.7.11 DAC sample and hold time register (DAC_SHHR) . . . 1275
28.7.12 DAC sample and hold refresh time register (DAC_SHRR) . . . 1275
28.7.13 DAC channel1 sawtooth register (DAC_STR1) . . . 1276
28.7.14 DAC sawtooth mode register (DAC_STMODR) . . . 1276
28.7.15 DAC extended sawtooth reset control register (DAC_STEXTRST) . 1277

28.7.16 DAC extended sawtooth minimum and maximum value register (DAC_STEXTMM1) . . . 1277
28.7.17 DAC option register (DAC_OR) . . . 1278
28.7.18 DAC register map . . . 1278

29 Voltage reference buffer (VREFBUF) . . . 1281
29.1 VREFBUF introduction . . . 1281
29.2 VREFBUF implementation . . . 1281
29.3 VREFBUF functional description . . . 1281
29.4 VREFBUF trimming . . . 1282
29.5 VREFBUF registers . . . 1283
29.5.1 VREFBUF control and status register (VREFBUF_CSR) . . . 1283
29.5.2 VREFBUF calibration control register (VREFBUF_CCR) . . . 1284
29.5.3 VREFBUF register map . . . 1285

30 Comparator (COMP) . . . 1286
30.1 COMP introduction . . . 1286
30.2 COMP main features . . . 1286
30.3 COMP functional description . . . 1287
30.3.1 COMP simplified block diagram . . . 1287
30.3.2 COMP pins and internal signals . . . 1287
30.3.3 COMP reset and clocks . . . 1289
30.3.4 Comparator LOCK mechanism . . . 1289
30.3.5 Window comparator . . . 1289
30.3.6 Hysteresis . . . 1290
30.3.7 Deglitcher feature . . . 1291
30.3.8 Digital signal filter . . . 1291
30.3.9 Comparator output blanking function . . . 1292
30.3.10 COMP startup stabilization time . . . 1292
30.4 COMP low-power modes . . . 1293
30.5 COMP interrupts . . . 1293
30.6 COMP registers . . . 1293
30.6.1 Comparator x configuration register (COMPx_CSR) . . . 1293
30.6.2 Comparator x filter configuration register (COMPx_FLTR) . . . 1295
30.6.3 COMPx register map . . . 1296

31 Operational amplifier (OPAMP) 1297

31.1 Introduction 1297

31.2 OPAMP main features 1297

31.3 OPAMP implementation 1297

31.4 OPAMP functional description 1299

31.4.1 OPAMP block diagram 1299

31.4.2 OPAMP pins and internal signals 1300

31.4.3 OPAMP reset and clocks 1301

31.4.4 OPAMP output redirection to an internal ADC/COMP channel 1302

31.4.5 OPAMP initial configuration 1302

31.4.6 OPAMP operational amplifier speed 1302

31.4.7 OPAMP enable 1302

31.4.8 OPAMP operating modes 1302

31.4.9 OPAMP programmable gain 1309

31.4.10 OPAMP operational amplifier offset compensation 1310

31.4.11 OPAMP calibration 1310

31.4.12 Timer-controlled operation 1312

31.5 OPAMP in low-power modes 1313

31.6 OPAMP registers 1314

31.6.1 OPAMP control/status register (OPAMP_CSR) 1314

31.6.2 OPAMP timer-controlled mode register (OPAMP_TCMR) 1317

31.6.3 OPAMP register map 1320

32 Coupling and chaining bridge (CCB) 1321

32.1 CCB introduction 1321

32.2 CCB main features 1321

32.3 CCB implementation 1322

32.4 CCB functional description 1322

32.4.1 CCB block diagram 1322

32.4.2 CCB internal signals 1323

32.4.3 CCB reset and clocks 1323

32.4.4 CCB coupling and chaining 1323

32.4.5 CCB error management 1326

32.5 CCB coupling and chaining operations 1326

32.5.1 Preliminary steps before any protected operation 1328

32.5.2 ECDSA signature 1329

32.5.3 ECC scalar multiplication 1336
32.5.4 RSA modular exponentiation 1339
32.6 CCB interrupts 1341
32.7 CCB registers 1342
32.7.1 CCB control register (CCB_CR) 1342
32.7.2 CCB status register (CCB_SR) 1343
32.7.3 CCB reference tag register (CCB_REFTAGRx) 1345
32.7.4 CCB register map 1346

33 True random number generator (RNG) 1347
33.1 RNG introduction 1347
33.2 RNG main features 1347
33.3 RNG functional description 1348
33.3.1 RNG block diagram 1348
33.3.2 RNG internal signals 1348
33.3.3 Random number generation 1349
33.3.4 RNG initialization 1351
33.3.5 RNG operation 1352
33.3.6 RNG clocking 1354
33.3.7 Error management 1354
33.3.8 RNG low-power use 1356
33.4 RNG interrupts 1356
33.5 RNG processing time 1357
33.6 RNG entropy source validation 1357
33.6.1 Introduction 1357
33.6.2 Validation conditions 1357
33.7 RNG registers 1359
33.7.1 RNG control register (RNG_CR) 1359
33.7.2 RNG status register (RNG_SR) 1361
33.7.3 RNG data register (RNG_DR) 1362
33.7.4 RNG noise source control register (RNG_NSCR) 1363
33.7.5 RNG health test control register x (RNG_HTCRx) 1363
33.7.6 RNG health test status register 0 (RNG_HTSR0) 1364
33.7.7 RNG health test status register 1 (RNG_HTSR1) 1364
33.7.8 RNG noise source mask register (RNG_NSMR) 1365
33.7.9 RNG register map 1366

34 AES hardware accelerator (AES) 1367
34.1 AES introduction 1367
34.2 AES main features 1367
34.3 AES implementation 1368
34.4 AES functional description 1368
34.4.1 AES block diagram 1368
34.4.2 AES internal signals 1369
34.4.3 AES reset and clocks 1369
34.4.4 AES symmetric cipher implementation 1369
34.4.5 AES encryption or decryption typical usage 1370
34.4.6 AES authenticated encryption, decryption, and cipher-based message authentication 1373
34.4.7 AES ciphertext stealing and data padding 1373
34.4.8 AES suspend and resume operations 1374
34.4.9 AES basic chaining modes (ECB, CBC) 1374
34.4.10 AES counter (CTR) mode 1378
34.4.11 AES Galois/counter mode (GCM) 1380
34.4.12 AES Galois message authentication code (GMAC) 1385
34.4.13 AES counter with CBC-MAC (CCM) 1386
34.4.14 AES key sharing with secure AES co-processor 1391
34.4.15 AES data registers and data swapping 1392
34.4.16 AES key registers 1394
34.4.17 AES initialization vector registers 1394
34.4.18 AES error management 1395
34.5 AES interrupts 1396
34.6 AES DMA requests 1396
34.7 AES processing latency 1397
34.8 AES registers 1399
34.8.1 AES control register (AES_CR) 1399
34.8.2 AES status register (AES_SR) 1401
34.8.3 AES data input register (AES_DINR) 1402
34.8.4 AES data output register (AES_DOUTR) 1403
34.8.5 AES key register 0 (AES_KEYR0) 1403
34.8.6 AES key register 1 (AES_KEYR1) 1404
34.8.7 AES key register 2 (AES_KEYR2) 1404
34.8.8 AES key register 3 (AES_KEYR3) 1404

34.8.9 AES initialization vector register 0 (AES_IVR0) 1405
34.8.10 AES initialization vector register 1 (AES_IVR1) 1405
34.8.11 AES initialization vector register 2 (AES_IVR2) 1405
34.8.12 AES initialization vector register 3 (AES_IVR3) 1406
34.8.13 AES key register 4 (AES_KEYR4) 1406
34.8.14 AES key register 5 (AES_KEYR5) 1406
34.8.15 AES key register 6 (AES_KEYR6) 1407
34.8.16 AES key register 7 (AES_KEYR7) 1407
34.8.17 AES suspend registers (AES_SUSPRx) 1407
34.8.18 AES interrupt enable register (AES_IER) 1408
34.8.19 AES interrupt status register (AES_ISR) 1409
34.8.20 AES interrupt clear register (AES_ICR) 1410
34.8.21 AES register map 1410

35 Secure AES coprocessor (SAES) 1413

35.1 SAES introduction 1413
35.2 SAES main features 1413
35.3 SAES implementation 1414
35.4 SAES functional description 1414
35.4.1 SAES block diagram 1414
35.4.2 SAES internal signals 1415
35.4.3 SAES reset and clocks 1416
35.4.4 SAES symmetric cipher implementation 1416
35.4.5 SAES encryption or decryption typical usage 1417
35.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication 1419
35.4.7 SAES ciphertext stealing and data padding 1420
35.4.8 SAES suspend and resume operations 1420
35.4.9 SAES basic chaining modes (ECB, CBC) 1421
35.4.10 SAES counter (CTR) mode 1425
35.4.11 SAES Galois/counter mode (GCM) 1427
35.4.12 SAES Galois message authentication code (GMAC) 1431
35.4.13 SAES counter with CBC-MAC (CCM) 1433
35.4.14 SAES operation with wrapped keys 1438
35.4.15 SAES operation with shared keys 1442
35.4.16 SAES data registers and data swapping 1443
35.4.17 SAES key registers 1446

35.4.18 SAES initialization vector registers . . . 1447
35.4.19 SAES error management . . . 1448
35.5 SAES interrupts . . . 1450
35.6 SAES DMA requests . . . 1450
35.7 SAES processing latency . . . 1451
35.8 SAES registers . . . 1453
35.8.1 SAES control register (SAES_CR) . . . 1453
35.8.2 SAES status register (SAES_SR) . . . 1456
35.8.3 SAES data input register (SAES_DINR) . . . 1457
35.8.4 SAES data output register (SAES_DOUTR) . . . 1458
35.8.5 SAES key register 0 (SAES_KEYR0) . . . 1458
35.8.6 SAES key register 1 (SAES_KEYR1) . . . 1459
35.8.7 SAES key register 2 (SAES_KEYR2) . . . 1459
35.8.8 SAES key register 3 (SAES_KEYR3) . . . 1459
35.8.9 SAES initialization vector register 0 (SAES_IVR0) . . . 1460
35.8.10 SAES initialization vector register 1 (SAES_IVR1) . . . 1460
35.8.11 SAES initialization vector register 2 (SAES_IVR2) . . . 1460
35.8.12 SAES initialization vector register 3 (SAES_IVR3) . . . 1461
35.8.13 SAES key register 4 (SAES_KEYR4) . . . 1461
35.8.14 SAES key register 5 (SAES_KEYR5) . . . 1461
35.8.15 SAES key register 6 (SAES_KEYR6) . . . 1462
35.8.16 SAES key register 7 (SAES_KEYR7) . . . 1462
35.8.17 SAES suspend registers (SAES_SUSPRx) . . . 1462
35.8.18 SAES interrupt enable register (SAES_IER) . . . 1463
35.8.19 SAES interrupt status register (SAES_ISR) . . . 1464
35.8.20 SAES interrupt clear register (SAES_ICR) . . . 1465
35.8.21 SAES register map . . . 1466

36 Hash processor (HASH) . . . 1468
36.1 HASH introduction . . . 1468
36.2 HASH main features . . . 1468
36.3 HASH implementation . . . 1469
36.4 HASH functional description . . . 1469
36.4.1 HASH block diagram . . . 1469
36.4.2 HASH internal signals . . . 1469
36.4.3 About secure hash algorithms . . . 1470

36.4.4 Message data feeding 1470
36.4.5 Message digest computing 1471
36.4.6 Message padding 1473
36.4.7 HMAC operation 1475
36.4.8 HASH suspend/resume operations 1476
36.4.9 HASH DMA interface 1478
36.4.10 HASH error management 1479
36.4.11 HASH processing time 1479
36.5 HASH interrupts 1480
36.6 HASH registers 1480
36.6.1 HASH control register (HASH_CR) 1480
36.6.2 HASH data input register (HASH_DIN) 1482
36.6.3 HASH start register (HASH_STR) 1483
36.6.4 HASH digest registers 1484
36.6.5 HASH interrupt enable register (HASH_IMR) 1486
36.6.6 HASH status register (HASH_SR) 1486
36.6.7 HASH context swap registers 1487
36.6.8 HASH register map 1488

37 Public key accelerator (PKA) 1490
37.1 PKA introduction 1490
37.2 PKA main features 1490
37.3 PKA functional description 1491
37.3.1 PKA block diagram 1491
37.3.2 PKA internal signals 1491
37.3.3 PKA reset and clocks 1491
37.3.4 PKA public key acceleration 1491
37.3.5 Typical applications for PKA 1493
37.3.6 PKA procedure to perform an operation 1495
37.3.7 PKA error management 1496
37.4 PKA operating modes 1497
37.4.1 Introduction 1497
37.4.2 Montgomery parameter computation 1498
37.4.3 Modular addition 1498
37.4.4 Modular subtraction 1499
37.4.5 Modular and Montgomery multiplication 1499

37.4.6 Modular exponentiation 1500
37.4.7 Modular inversion 1501
37.4.8 Modular reduction 1502
37.4.9 Arithmetic addition 1502
37.4.10 Arithmetic subtraction 1503
37.4.11 Arithmetic multiplication 1503
37.4.12 Arithmetic comparison 1503
37.4.13 RSA CRT exponentiation 1504
37.4.14 Point on elliptic curve Fp check 1505
37.4.15 ECC Fp scalar multiplication 1505
37.4.16 ECDSA sign 1506
37.4.17 ECDSA verification 1508
37.4.18 ECC complete addition 1509
37.4.19 ECC double base ladder 1509
37.4.20 ECC projective to affine 1510
37.5 Example of configurations and processing times 1511
37.5.1 Supported elliptic curves 1511
37.5.2 Computation times 1513
37.6 PKA interrupts 1515
37.7 PKA registers 1516
37.7.1 PKA control register (PKA_CR) 1516
37.7.2 PKA status register (PKA_SR) 1518
37.7.3 PKA clear flag register (PKA_CLRFR) 1519
37.7.4 PKA RAM 1519
37.7.5 PKA register map 1520

38 Public key secure accelerator (PKA) 1521
38.1 PKA introduction 1521
38.2 PKA main features 1521
38.3 PKA functional description 1522
38.3.1 PKA block diagram 1522
38.3.2 PKA internal signals 1522
38.3.3 PKA reset and clocks 1523
38.3.4 PKA public key acceleration 1523
38.3.5 Typical applications for PKA 1525
38.3.6 PKA procedure to perform an operation 1527

38.3.7 Private keys protection using CCB 1528
38.3.8 PKA error management 1532
38.4 PKA operating modes 1532
38.4.1 Introduction 1532
38.4.2 Montgomery parameter computation 1534
38.4.3 Modular addition 1534
38.4.4 Modular subtraction 1535
38.4.5 Modular and Montgomery multiplication 1535
38.4.6 Modular exponentiation 1536
38.4.7 Modular inversion 1538
38.4.8 Modular reduction 1538
38.4.9 Arithmetic addition 1539
38.4.10 Arithmetic subtraction 1539
38.4.11 Arithmetic multiplication 1540
38.4.12 Arithmetic comparison 1540
38.4.13 RSA CRT exponentiation 1540
38.4.14 Point on elliptic curve Fp check 1541
38.4.15 ECC Fp scalar multiplication 1542
38.4.16 ECDSA sign 1543
38.4.17 ECDSA verification 1545
38.4.18 ECC complete addition 1546
38.4.19 ECC double base ladder 1546
38.4.20 ECC projective to affine 1547
38.5 Example of configurations and processing times 1548
38.5.1 Supported elliptic curves 1548
38.5.2 Computation times 1550
38.6 PKA interrupts 1552
38.7 PKA registers 1553
38.7.1 PKA control register (PKA_CR) 1553
38.7.2 PKA status register (PKA_SR) 1555
38.7.3 PKA clear flag register (PKA_CLRFR) 1557
38.7.4 PKA RAM 1558
38.7.5 PKA register map 1559

39 On-the-fly decryption engine (OTFDEC) 1560

39.1 OTFDEC introduction 1560

39.2 OTFDEC main features 1560
39.3 OTFDEC functional description 1561
39.3.1 OTFDEC block diagram 1561
39.3.2 OTFDEC internal signals 1561
39.3.3 OTFDEC on-the-fly decryption 1562
39.3.4 OTFDEC usage of AES in counter mode decryption 1563
39.3.5 Flow control management 1564
39.3.6 OTFDEC error management 1564
39.4 OTFDEC interrupts 1565
39.5 OTFDEC application information 1565
39.5.1 OTFDEC initialization process 1565
39.5.2 OTFDEC and power management 1567
39.5.3 Encrypting for OTFDEC 1567
39.5.4 OTFDEC key CRC source code 1568
39.6 OTFDEC registers 1569
39.6.1 OTFDEC control register (OTFDEC_CR) 1569
39.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR) 1570
39.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR) 1570
39.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR) 1572
39.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR) . 1572
39.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0) 1573
39.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1) 1574
39.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0) 1574
39.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1) 1575
39.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2) 1575
39.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3) 1576
39.6.12 OTFDEC interrupt status register (OTFDEC_ISR) 1576
39.6.13 OTFDEC interrupt clear register (OTFDEC_ICR) 1577
39.6.14 OTFDEC interrupt enable register (OTFDEC_IER) 1578
39.6.15 OTFDEC register map 1579

40 High-resolution timer (HRTIM) 1583
40.1 Introduction 1583
40.2 Main features 1584
40.3 Functional description 1585

40.3.1 General description . . . 1585
40.3.2 HRTIM pins and internal signals . . . 1587
40.3.3 Clocks . . . 1593
40.3.4 Timer A..F timing units . . . 1597
40.3.5 Master timer . . . 1628
40.3.6 Up-down counting mode . . . 1630
40.3.7 Set/reset events priorities and narrow pulses management . . . 1637
40.3.8 External events global conditioning . . . 1640
40.3.9 External event filtering in timing units . . . 1645
40.3.10 Delayed protection . . . 1653
40.3.11 Register preload and update management . . . 1659
40.3.12 PWM mode with “greater than” comparison . . . 1663
40.3.13 Events propagation within or across multiple timers . . . 1664
40.3.14 Output management . . . 1669
40.3.15 Burst mode controller . . . 1671
40.3.16 Chopper . . . 1679
40.3.17 Fault protection . . . 1680
40.3.18 Auxiliary outputs . . . 1685
40.3.19 Synchronizing the HRTIM with other timers or HRTIM instances . . . 1688
40.3.20 ADC triggers . . . 1691
40.3.21 DAC triggers . . . 1695
40.3.22 Interrupts . . . 1699
40.3.23 DMA . . . 1701
40.3.24 HRTIM initialization . . . 1705
40.3.25 Debug . . . 1706
40.4 Application use cases . . . 1707
40.4.1 Buck converter . . . 1707
40.4.2 Buck converter with synchronous rectification . . . 1708
40.4.3 Multiphase converters . . . 1709
40.4.4 Transition mode power factor correction . . . 1711
40.5 HRTIM registers . . . 1713
40.5.1 HRTIM master timer control register (HRTIM_MCR) . . . 1713
40.5.2 HRTIM master timer interrupt status register (HRTIM_MISR) . . . 1716
40.5.3 HRTIM master timer interrupt clear register (HRTIM_MICR) . . . 1717
40.5.4 HRTIM master timer DMA interrupt enable register (HRTIM_MDIER) . . . 1718
40.5.5 HRTIM master timer counter register (HRTIM_MCNTR) . . . 1720

40.5.6 HRTIM master timer period register (HRTIM_MPER) 1720
40.5.7 HRTIM master timer repetition register (HRTIM_MREP) 1721
40.5.8 HRTIM master timer compare 1 register (HRTIM_MCMP1R) 1721
40.5.9 HRTIM master timer compare 2 register (HRTIM_MCMP2R) 1722
40.5.10 HRTIM master timer compare 3 register (HRTIM_MCMP3R) 1722
40.5.11 HRTIM master timer compare 4 register (HRTIM_MCMP4R) 1723
40.5.12 HRTIM timer x control register (HRTIM_TIMxCR) (x = A to F) 1724
40.5.13 HRTIM timer x interrupt status register (HRTIM_TIMxISR) (x = A to F) 1728
40.5.14 HRTIM timer x interrupt clear register (HRTIM_TIMxICR) (x = A to F) 1730
40.5.15 HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F) 1732
40.5.16 HRTIM timer x counter register (HRTIM_CNTxR) (x = A to F) 1735
40.5.17 HRTIM timer x period register (HRTIM_PERxR) (x = A to F) 1735
40.5.18 HRTIM timer x repetition register (HRTIM_REPxR) (x = A to F) 1737
40.5.19 HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F) . . 1737
40.5.20 HRTIM timer x compare 1 compound register (HRTIM_CMP1CxR) (x = A to F) 1739
40.5.21 HRTIM timer x compare 2 register (HRTIM_CMP2xR) (x = A to F) . . 1739
40.5.22 HRTIM timer x compare 3 register (HRTIM_CMP3xR) (x = A to F) . . 1741
40.5.23 HRTIM timer x compare 4 register (HRTIM_CMP4xR) (x = A to F) . . 1742
40.5.24 HRTIM timer x capture 1 register (HRTIM_CPT1xR) (x = A to F) . . . 1743
40.5.25 HRTIM timer x capture 2 register (HRTIM_CPT2xR) (x = A to F) 1744
40.5.26 HRTIM timer x deadtime register (HRTIM_DTxR) (x = A to F) 1745
40.5.27 HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F) . 1747
40.5.28 HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F) 1750
40.5.29 HRTIM timer x output 2 set register (HRTIM_SETx2R) (x = A to F) . 1752
40.5.30 HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F) 1755
40.5.31 HRTIM timer x external event filtering register 1 (HRTIM_EEFxR1) (x = A to F) 1758
40.5.32 HRTIM timer x external event filtering register 2 (HRTIM_EEFxR2) (x = A to F) 1761
40.5.33 HRTIM timer A reset register (HRTIM_RSTAR) 1762
40.5.34 HRTIM timer B reset register (HRTIM_RSTBR) 1764
40.5.35 HRTIM timer C reset register (HRTIM_RSTCR) 1766
40.5.36 HRTIM timer D reset register (HRTIM_RSTDR) 1768
40.5.37 HRTIM timer E reset register (HRTIM_RSTER) 1770

40.5.38 HRTIM timer F reset register (HRTIM_RSTFR) 1772
40.5.39 HRTIM timer x chopper register (HRTIM_CHPxR) (x = A to F) 1774
40.5.40 HRTIM timer A capture 1 control register (HRTIM_CPT1ACR) 1775
40.5.41 HRTIM timer B capture 1 control register (HRTIM_CPT1BCR) 1777
40.5.42 HRTIM timer C capture 1 control register (HRTIM_CPT1CCR) 1779
40.5.43 HRTIM timer D capture 1 control register (HRTIM_CPT1DCR) 1781
40.5.44 HRTIM timer E capture 1 control register (HRTIM_CPT1ECR) 1783
40.5.45 HRTIM timer F capture 1 control register (HRTIM_CPT1FCR) 1785
40.5.46 HRTIM timer A capture 2 control register (HRTIM_CPT2ACR) 1787
40.5.47 HRTIM timer B capture 2 control register (HRTIM_CPT2BCR) 1789
40.5.48 HRTIM timer C capture 2 control register (HRTIM_CPT2CCR) 1791
40.5.49 HRTIM timer D capture 2 control register (HRTIM_CPT2DCR) 1793
40.5.50 HRTIM timer E capture 2 control register (HRTIM_CPT2ECR) 1795
40.5.51 HRTIM timer F capture 2 control register (HRTIM_CPT2FCR) 1797
40.5.52 HRTIM timer x output register (HRTIM_OUTxR) (x = A to F) 1799
40.5.53 HRTIM timer x fault register (HRTIM_FLTxR) (x = A to F) 1802
40.5.54 HRTIM timer x control register 2 (HRTIM_TIMxCR2) (x = A to F) . . . 1804
40.5.55 HRTIM timer x external event filtering register 3 (HRTIM_TIMxEEFR3) (x = A to F) 1808
40.5.56 HRTIM timer x external event selection register (HRTIM_EESELxR) (x = A to F) 1810
40.5.57 HRTIM timer x crossbar override trigger register (HRTIM_COTRGxR) (x = A to F) 1813
40.5.58 HRTIM timer x crossbar override forcing event register (HRTIM_COFExR) (x = A to F) 1816
40.5.59 HRTIM timer x crossbar override delayed fault event register (HRTIM_CODExR) (x = A to F) 1819
40.5.60 HRTIM control register 1 (HRTIM_CR1) 1820
40.5.61 HRTIM control register 2 (HRTIM_CR2) 1822
40.5.62 HRTIM interrupt status register (HRTIM_ISR) 1824
40.5.63 HRTIM interrupt clear register (HRTIM_ICR) 1825
40.5.64 HRTIM interrupt enable register (HRTIM_IER) 1826
40.5.65 HRTIM output enable register (HRTIM_OENR) 1827
40.5.66 HRTIM output disable register (HRTIM_ODISR) 1828
40.5.67 HRTIM output disable status register (HRTIM_ODSR) 1829
40.5.68 HRTIM burst mode control register (HRTIM_BMCR) 1830
40.5.69 HRTIM burst mode trigger register (HRTIM_BMTRGR) 1832
40.5.70 HRTIM burst mode compare register (HRTIM_BMCMPR) 1834

40.5.71 HRTIM burst mode period register (HRTIM_BMPER) 1834
40.5.72 HRTIM timer external event control register 1 (HRTIM_EECR1) . . . 1835
40.5.73 HRTIM timer external event control register 2 (HRTIM_EECR2) . . . 1837
40.5.74 HRTIM timer external event control register 3 (HRTIM_EECR3) . . . 1838
40.5.75 HRTIM ADC trigger 1 register (HRTIM_ADC1R) 1839
40.5.76 HRTIM ADC trigger 2 register (HRTIM_ADC2R) 1842
40.5.77 HRTIM ADC trigger 3 register (HRTIM_ADC3R) 1844
40.5.78 HRTIM ADC trigger 4 register (HRTIM_ADC4R) 1846
40.5.79 HRTIM DLL control register (HRTIM_DLLCR) 1848
40.5.80 HRTIM fault input register 1 (HRTIM_FLTINR1) 1849
40.5.81 HRTIM fault input register 2 (HRTIM_FLTINR2) 1851
40.5.82 HRTIM burst DMA master timer update register (HRTIM_BDMUPR) 1853
40.5.83 HRTIM burst DMA timer x update register (HRTIM_BDTxUPR) (x = A to F) 1854
40.5.84 HRTIM burst DMA data register (HRTIM_BDMADR) 1856
40.5.85 HRTIM ADC extended trigger register (HRTIM_ADCER) 1857
40.5.86 HRTIM ADC trigger update register (HRTIM_ADCUR) 1860
40.5.87 HRTIM ADC post scaler register 1 (HRTIM_ADCPS1) 1861
40.5.88 HRTIM ADC post scaler register 2 (HRTIM_ADCPS2) 1862
40.5.89 HRTIM fault input register 3 (HRTIM_FLTINR3) 1863
40.5.90 HRTIM fault input register 4 (HRTIM_FLTINR4) 1865
40.5.91 HRTIM fault input register 5 (HRTIM_FLTINR5) 1867
40.5.92 HRTIM timer external event control register 4 (HRTIM_EECR4) . . . 1869
40.5.93 HRTIM timer external event control register 5 (HRTIM_EECR5) . . . 1871
40.5.94 HRTIM timer external event control register 6 (HRTIM_EECR6) . . . 1873
40.5.95 HRTIM timer external event control register 7 (HRTIM_EECR7) . . . 1874
40.5.96 HRTIM control register 3 (HRTIM_CR3) 1876
40.5.97 HRTIM option register (HRTIM_OR) 1876
40.5.98 HRTIM register map 1878

41 Advanced-control timers (TIM1/TIM8/TIM20) 1889

41.1 TIM1/TIM8/TIM20 introduction 1889
41.2 TIM1/TIM8/TIM20 main features 1889
41.3 TIM1/TIM8/TIM20 functional description 1890
41.3.1 Block diagram 1890
41.3.2 TIM1/TIM8/TIM20 pins and internal signals 1891
41.3.3 Time-base unit 1897

41.3.4 Counter modes 1900
41.3.5 Repetition counter 1912
41.3.6 External trigger input 1913
41.3.7 Clock selection 1914
41.3.8 Capture/compare channels 1918
41.3.9 Input capture mode 1921
41.3.10 PWM input mode 1922
41.3.11 Forced output mode 1923
41.3.12 Output compare mode 1923
41.3.13 PWM mode 1924
41.3.14 Asymmetric PWM mode 1933
41.3.15 Combined PWM mode 1936
41.3.16 Combined 3-phase PWM mode 1939
41.3.17 Complementary outputs and dead-time insertion 1940
41.3.18 Clock dividers for dead-time generators and digital filters 1943
41.3.19 Using the break function 1943
41.3.20 Delayed break 1949
41.3.21 Bidirectional break inputs 1951
41.3.22 Output idle state override 1953
41.3.23 Clearing the tim_ocxref signal on an external event 1954
41.3.24 6-step PWM generation 1957
41.3.25 One-pulse mode 1958
41.3.26 Retriggerable One-pulse mode 1960
41.3.27 Pulse on compare mode 1961
41.3.28 Encoder interface mode 1963
41.3.29 Encoder with built-in debouncer 1980
41.3.30 Direction bit output 1982
41.3.31 Clock drift measurement 1982
41.3.32 UIF bit remapping 1983
41.3.33 Timer input XOR function 1983
41.3.34 Interfacing with Hall sensors 1985
41.3.35 Timer synchronization 1986
41.3.36 ADC triggers 1991
41.3.37 ADC synchronization 1991
41.3.38 DMA burst mode 1992
41.3.39 TIM1/TIM8/TIM20 DMA requests 1993
41.3.40 Debug mode 1994

41.4 TIM1/TIM8/TIM20 low-power modes 1994
41.5 TIM1/TIM8/TIM20 interrupts 1994
41.6 TIM1/TIM8/TIM20 registers 1996
41.6.1 TIM control register 1 (TIM_CR1) 1996
41.6.2 TIM control register 2 (TIM_CR2) 1998
41.6.3 TIM slave mode control register (TIM_SMCR) 2003
41.6.4 TIM DMA/interrupt enable register (TIM_DIER) 2007
41.6.5 TIM status register (TIM_SR) 2008
41.6.6 TIM event generation register (TIM_EGR) 2012
41.6.7 TIM capture/compare mode register 1 (TIM_CCMR1) 2013
41.6.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2015
41.6.9 TIM capture/compare mode register 2 (TIM_CCMR2) 2019
41.6.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2) . . . 2021
41.6.11 TIM capture/compare enable register (TIM_CCER) 2024
41.6.12 TIM counter (TIM_CNT) 2028
41.6.13 TIM prescaler (TIM_PSC) 2028
41.6.14 TIM autoreload register (TIM_ARR) 2029
41.6.15 TIM repetition counter register (TIM_RCR) 2029
41.6.16 TIM capture/compare register 1 (TIM_CCR1) 2030
41.6.17 TIM capture/compare register 2 (TIM_CCR2) 2030
41.6.18 TIM capture/compare register 3 (TIM_CCR3) 2031
41.6.19 TIM capture/compare register 4 (TIM_CCR4) 2032
41.6.20 TIM break and dead-time register (TIM_BDTR) 2033
41.6.21 TIM capture/compare register 5 (TIM_CCR5) 2037
41.6.22 TIM capture/compare register 6 (TIM_CCR6) 2039
41.6.23 TIM capture/compare mode register 3 (TIM_CCMR3) 2040
41.6.24 TIM timer deadtime register 2 (TIM_DTR2) 2043
41.6.25 TIM timer encoder control register (TIM_ECR) 2044
41.6.26 TIM timer input selection register (TIM_TISEL) 2045
41.6.27 TIM alternate function option register 1 (TIM_AF1) 2046
41.6.28 TIM alternate function register 2 (TIM_AF2) 2051
41.6.29 TIM capture/compare register 7 (TIM_CCR7) 2055
41.6.30 TIM capture/compare mode register 4 (TIM_CCMR4) 2056
41.6.31 TIM multilevel protection register 1 (TIM_MPR1) 2058
41.6.32 TIM multilevel protection register 2 (TIM_MPR2) 2060
41.6.33 TIM output override register (TIM_OOR) 2062
41.6.34 TIM DMA control register (TIM_DCR) 2063

41.6.35 TIM DMA address for full transfer (TIM_DMAR) 2065
41.6.36 TIM1/TIM8/TIM20 register map 2065

42 General-purpose timers (TIM2/TIM3/TIM4/TIM5) 2069

42.1 TIM2/TIM3/TIM4/TIM5 introduction 2069
42.2 TIM2/TIM3/TIM4/TIM5 main features 2069
42.3 TIM2/TIM3/TIM4/TIM5 implementation 2070
42.4 TIM2/TIM3/TIM4/TIM5 functional description 2071
42.4.1 Block diagram 2071
42.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals 2072
42.4.3 Time-base unit 2077
42.4.4 Counter modes 2079
42.4.5 External trigger input 2091
42.4.6 Clock selection 2092
42.4.7 Capture/compare channels 2096
42.4.8 Input capture mode 2098
42.4.9 PWM input mode 2099
42.4.10 Forced output mode 2100
42.4.11 Output compare mode 2100
42.4.12 PWM mode 2102
42.4.13 Asymmetric PWM mode 2110
42.4.14 Combined PWM mode 2112
42.4.15 Clock dividers for dead-time generators and digital filters 2113
42.4.16 Clearing the tim_ocxref signal on an external event 2114
42.4.17 One-pulse mode 2117
42.4.18 Retriggerable one-pulse mode 2119
42.4.19 Pulse on compare mode 2120
42.4.20 Encoder interface mode 2122
42.4.21 Encoder with built-in debouncer 2140
42.4.22 Direction bit output 2142
42.4.23 Clock drift measurement 2142
42.4.24 UIF bit remapping 2143
42.4.25 Timer input XOR function 2143
42.4.26 Timers and external trigger synchronization 2145
42.4.27 Timer synchronization 2148
42.4.28 ADC triggers 2153
42.4.29 ADC synchronization 2155

42.4.30 DMA burst mode 2155
42.4.31 TIM2/TIM3/TIM4/TIM5 DMA requests 2156
42.4.32 Debug mode 2157
42.4.33 TIM2/TIM3/TIM4/TIM5 low-power modes 2157
42.4.34 TIM2/TIM3/TIM4/TIM5 interrupts 2158
42.5 TIM2/TIM3/TIM4/TIM5 registers 2159
42.5.1 TIM control register 1 (TIM_CR1) 2159
42.5.2 TIM control register 2 (TIM_CR2) 2161
42.5.3 TIM slave mode control register (TIM_SMCR) 2164
42.5.4 TIM DMA/interrupt enable register (TIM_DIER) 2168
42.5.5 TIM status register (TIM_SR) 2169
42.5.6 TIM event generation register (TIM_EGR) 2172
42.5.7 TIM capture/compare mode register 1 (TIM_CCMR1) 2173
42.5.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2175
42.5.9 TIM capture/compare mode register 2 (TIM_CCMR2) 2177
42.5.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2) . . . 2178
42.5.11 TIM capture/compare enable register (TIM_CCER) 2181
42.5.12 TIM counter (TIM_CNT) 2183
42.5.13 TIM prescaler (TIM_PSC) 2183
42.5.14 TIM autoreload register (TIM_ARR) 2184
42.5.15 TIM capture/compare register 1 (TIM_CCR1) 2184
42.5.16 TIM capture/compare register 2 (TIM_CCR2) 2185
42.5.17 TIM capture/compare register 3 (TIM_CCR3) 2186
42.5.18 TIM capture/compare register 4 (TIM_CCR4) 2187
42.5.19 TIM timer encoder control register (TIM_ECR) 2188
42.5.20 TIM timer input selection register (TIM_TISEL) 2189
42.5.21 TIM alternate function register 1 (TIM_AF1) 2190
42.5.22 TIM alternate function register 2 (TIM_AF2) 2191
42.5.23 TIM DMA control register (TIM_DCR) 2192
42.5.24 TIM DMA address for full transfer (TIM_DMAR) 2193
42.5.25 TIM2/TIM3/TIM4/TIM5 register map 2194

43 Basic timers (TIM6/TIM7/TIM26/TIM27) 2197
43.1 TIM6/TIM7/TIM26/TIM27 introduction 2197
43.2 TIM6/TIM7/TIM26/TIM27 main features 2197
43.3 TIM6/TIM7/TIM26/TIM27 functional description 2198
43.3.1 TIM6/TIM7/TIM26/TIM27 block diagram 2198

43.3.2 TIM6/TIM7/TIM26/TIM27 internal signals 2198
43.3.3 TIM6/TIM7/TIM26/TIM27 clocks 2199
43.3.4 Time-base unit 2199
43.3.5 Counting mode 2201
43.3.6 UIF bit remapping 2208
43.3.7 ADC triggers 2209
43.3.8 ADC synchronization 2209
43.3.9 TIM6/TIM7/TIM26/TIM27 DMA requests 2210
43.3.10 Debug mode 2210
43.3.11 TIM6/TIM7/TIM26/TIM27 low-power modes 2210
43.3.12 TIM6/TIM7/TIM26/TIM27 interrupts 2210
43.4 TIM6/TIM7/TIM26/TIM27 registers 2211
43.4.1 TIM control register 1 (TIM_CR1) 2211
43.4.2 TIM control register 2 (TIM_CR2) 2213
43.4.3 TIM DMA/interrupt enable register (TIM_DIER) 2213
43.4.4 TIM status register (TIM_SR) 2215
43.4.5 TIM event generation register (TIM_EGR) 2215
43.4.6 TIM counter (TIM_CNT) 2216
43.4.7 TIM prescaler (TIM_PSC) 2216
43.4.8 TIM autoreload register (TIM_ARR) 2217
43.4.9 TIM6/TIM7/TIM26/TIM27 register map 2218

44 General-purpose timers (TIM9/TIM10/TIM11/TIM12/TIM13/TIM14) . 2219

44.1 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 introduction 2219
44.2 TIM9/TIM12 main features 2219
44.3 TIM10/TIM11/TIM13/TIM14 main features 2220
44.4 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 functional description 2221
44.4.1 Block diagram 2221
44.4.2 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals . . 2222
44.4.3 Time-base unit 2225
44.4.4 Counter modes 2227
44.4.5 Clock selection 2230
44.4.6 Capture/compare channels 2232
44.4.7 Input capture mode 2234
44.4.8 PWM input mode (TIM9/TIM12 only) 2235
44.4.9 Forced output mode 2236
44.4.10 Output compare mode 2237

44.4.11 PWM mode 2238
44.4.12 Combined PWM mode (TIM9/TIM12 only) 2243
44.4.13 Clock dividers for digital filters 2244
44.4.14 One-pulse mode 2245
44.4.15 Retriggerable one pulse mode (TIM9/TIM12 only) 2246
44.4.16 UIF bit remapping 2247
44.4.17 Timer input XOR function 2247
44.4.18 TIM9/TIM12 external trigger synchronization 2249
44.4.19 Slave mode – combined reset + trigger mode 2252
44.4.20 Slave mode – combined reset + gated mode 2252
44.4.21 Timer synchronization (TIM9/TIM12 only) 2252
44.4.22 Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only) 2252
44.4.23 ADC triggers (TIM9/TIM12 only) 2252
44.4.24 ADC synchronization (TIM9/TIM12 only) 2253
44.4.25 Debug mode 2254
44.5 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 low-power modes 2254
44.6 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 interrupts 2254
44.7 TIM9/TIM12 registers 2254
44.7.1 TIM control register 1 (TIM_CR1) 2255
44.7.2 TIM control register 2 (TIM_CR2) 2256
44.7.3 TIM slave mode control register (TIM_SMCR) 2258
44.7.4 TIM interrupt enable register (TIM_DIER) 2260
44.7.5 TIM status register (TIM_SR) 2261
44.7.6 TIM event generation register (TIM_EGR) 2262
44.7.7 TIM capture/compare mode register 1 (TIM_CCMR1) 2263
44.7.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2264
44.7.9 TIM capture/compare enable register (TIM_CCER) 2267
44.7.10 TIM counter (TIM_CNT) 2268
44.7.11 TIM prescaler (TIM_PSC) 2269
44.7.12 TIM autoreload register (TIM_ARR) 2269
44.7.13 TIM capture/compare register 1 (TIM_CCR1) 2270
44.7.14 TIM capture/compare register 2 (TIM_CCR2) 2270
44.7.15 TIM timer input selection register (TIM_TISEL) 2271
44.7.16 TIM9/TIM12 register map 2272
44.8 TIM10/TIM11/TIM13/TIM14 registers 2274
44.8.1 TIM control register 1 (TIM_CR1) 2274

44.8.2 TIM interrupt enable register (TIM_DIER) 2275
44.8.3 TIM status register (TIM_SR) 2276
44.8.4 TIM event generation register (TIM_EGR) 2277
44.8.5 TIM capture/compare mode register 1 (TIM_CCMR1) 2278
44.8.6 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2279
44.8.7 TIM capture/compare enable register (TIM_CCER) 2280
44.8.8 TIM counter (TIM_CNT) 2281
44.8.9 TIM prescaler (TIM_PSC) 2282
44.8.10 TIM autoreload register (TIM_ARR) 2282
44.8.11 TIM capture/compare register 1 (TIM_CCR1) 2283
44.8.12 TIM timer input selection register (TIM_TISEL) 2283
44.8.13 TIM10/TIM11/TIM13/TIM14 register map 2284

45 General purpose timers (TIM15/TIM16/TIM17) 2286

45.1 TIM15/TIM16/TIM17 introduction 2286
45.2 TIM15 main features 2286
45.3 TIM16/TIM17 main features 2287
45.4 TIM15/TIM16/TIM17 functional description 2288
45.4.1 Block diagram 2288
45.4.2 TIM15/TIM16/TIM17 pins and internal signals 2289
45.4.3 Time-base unit 2294
45.4.4 Counter modes 2296
45.4.5 Repetition counter 2308
45.4.6 Clock selection 2310
45.4.7 Capture/compare channels 2312
45.4.8 Input capture mode 2314
45.4.9 PWM input mode (only for TIM15) 2315
45.4.10 Forced output mode 2316
45.4.11 Output compare mode 2317
45.4.12 PWM mode 2318
45.4.13 Asymmetric PWM mode (TIM15 only) 2325
45.4.14 Combined PWM mode (TIM15 only) 2326
45.4.15 Complementary outputs and dead-time insertion 2327
45.4.16 Clock dividers for dead-time generators and digital filters 2330
45.4.17 Using the break function 2330
45.4.18 Delayed break 2335
45.4.19 Bidirectional break input 2337

45.4.20 Output idle state override . . . 2339
45.4.21 Clearing the tim_ocxref signal on an external event . . . 2339
45.4.22 6-step PWM generation . . . 2341
45.4.23 One-pulse mode . . . 2343
45.4.24 Retriggerable one pulse mode (TIM15 only) . . . 2344
45.4.25 Encoder interface mode . . . 2345
45.4.26 Encoder with built-in debouncer (TIM15 only) . . . 2352
45.4.27 Clock drift measurement . . . 2353
45.4.28 UIF bit remapping . . . 2354
45.4.29 Timer input XOR function (TIM15 only) . . . 2354
45.4.30 External trigger synchronization (TIM15 only) . . . 2356
45.4.31 Slave mode – combined reset + trigger mode (TIM15 only) . . . 2359
45.4.32 Slave mode – combined reset + gated mode (TIM15 only) . . . 2359
45.4.33 Timer synchronization (TIM15 only) . . . 2360
45.4.34 Using timer output as trigger for other timers (TIM16/TIM17 only) . . 2360
45.4.35 ADC triggers (TIM15 only) . . . 2360
45.4.36 ADC synchronization (TIM15 only) . . . 2361
45.4.37 DMA burst mode . . . 2361
45.4.38 TIM15/TIM16/TIM17 DMA requests . . . 2362
45.4.39 Debug mode . . . 2363
45.5 TIM15/TIM16/TIM17 low-power modes . . . 2363
45.6 TIM15/TIM16/TIM17 interrupts . . . 2363
45.7 TIM15 registers . . . 2365
45.7.1 TIM control register 1 (TIM_CR1) . . . 2365
45.7.2 TIM control register 2 (TIM_CR2) . . . 2367
45.7.3 TIM slave mode control register (TIM_SMCR) . . . 2370
45.7.4 TIM DMA/interrupt enable register (TIM_DIER) . . . 2373
45.7.5 TIM status register (TIM_SR) . . . 2374
45.7.6 TIM event generation register (TIM_EGR) . . . 2376
45.7.7 TIM capture/compare mode register 1 (TIM_CCMR1) . . . 2377
45.7.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2379
45.7.9 TIM capture/compare enable register (TIM_CCER) . . . 2382
45.7.10 TIM counter (TIM_CNT) . . . 2385
45.7.11 TIM prescaler (TIM_PSC) . . . 2385
45.7.12 TIM autoreload register (TIM_ARR) . . . 2385
45.7.13 TIM repetition counter register (TIM_RCR) . . . 2386
45.7.14 TIM capture/compare register 1 (TIM_CCR1) . . . 2386

45.7.15 TIM capture/compare register 2 (TIM_CCR2) 2388
45.7.16 TIM break and dead-time register (TIM_BDTR) 2388
45.7.17 TIM timer deadtime register 2 (TIM_DTR2) 2391
45.7.18 TIM input selection register (TIM_TISEL) 2392
45.7.19 TIM alternate function register 1 (TIM_AF1) 2393
45.7.20 TIM alternate function register 2 (TIM_AF2) 2397
45.7.21 TIM multilevel protection register 1 (TIM_MPR1) 2398
45.7.22 TIM multilevel protection register 2 (TIM_MPR2) 2398
45.7.23 TIM output override register (TIM_OOR) 2399
45.7.24 TIM DMA control register (TIM_DCR) 2400
45.7.25 TIM DMA address for full transfer (TIM_DMAR) 2401
45.7.26 TIM15 register map 2402
45.8 TIM16/TIM17 registers 2405
45.8.1 TIM control register 1 (TIM_CR1) 2405
45.8.2 TIM control register 2 (TIM_CR2) 2407
45.8.3 TIM DMA/interrupt enable register (TIM_DIER) 2408
45.8.4 TIM status register (TIM_SR) 2409
45.8.5 TIM event generation register (TIM_EGR) 2410
45.8.6 TIM capture/compare mode register 1 (TIM_CCMR1) 2411
45.8.7 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1) . . . 2412
45.8.8 TIM capture/compare enable register (TIM_CCER) 2414
45.8.9 TIM counter (TIM_CNT) 2417
45.8.10 TIM prescaler (TIM_PSC) 2417
45.8.11 TIM autoreload register (TIM_ARR) 2418
45.8.12 TIM repetition counter register (TIM_RCR) 2418
45.8.13 TIM capture/compare register 1 (TIM_CCR1) 2419
45.8.14 TIM break and dead-time register (TIM_BDTR) 2420
45.8.15 TIM timer deadtime register 2 (TIM_DTR2) 2423
45.8.16 TIM input selection register (TIM_TISEL) 2424
45.8.17 TIM alternate function register 1 (TIM_AF1) 2424
45.8.18 TIM alternate function register 2 (TIM_AF2) 2428
45.8.19 TIM multilevel protection register 1 (TIM_MPR1) 2429
45.8.20 TIM multilevel protection register 2 (TIM_MPR2) 2430
45.8.21 TIM output override register (TIM_OOR) 2430
45.8.22 TIM DMA control register (TIM_DCR) 2431
45.8.23 TIM DMA address for full transfer (TIM_DMAR) 2432
45.8.24 TIM16/TIM17 register map 2433

46 Low-power timer (LPTIM) 2436
46.1 LPTIM introduction 2436
46.2 LPTIM main features 2436
46.3 LPTIM implementation 2436
46.4 LPTIM functional description 2438
46.4.1 LPTIM block diagram 2438
46.4.2 LPTIM pins and internal signals 2438
46.4.3 LPTIM input and trigger mapping 2439
46.4.4 LPTIM reset and clocks 2441
46.4.5 Glitch filter 2441
46.4.6 Prescaler 2442
46.4.7 Trigger multiplexer 2442
46.4.8 Operating mode 2443
46.4.9 Timeout function 2445
46.4.10 Waveform generation 2445
46.4.11 Register update 2446
46.4.12 Counter mode 2447
46.4.13 Timer enable 2447
46.4.14 Timer counter reset 2448
46.4.15 Encoder mode 2448
46.4.16 Repetition counter 2450
46.4.17 Capture/compare channels 2451
46.4.18 Input capture mode 2452
46.4.19 PWM mode 2454
46.4.20 DMA requests 2456
46.4.21 Debug mode 2457
46.5 LPTIM low-power modes 2457
46.6 LPTIM interrupts 2457
46.7 LPTIM registers 2458
46.7.1 LPTIM1 interrupt and status register [alternate] (LPTIM1_ISR) 2459
46.7.2 LPTIM1 interrupt and status register [alternate] (LPTIM1_ISR) 2461
46.7.3 LPTIM1 interrupt clear register [alternate] (LPTIM1_ICR) 2463
46.7.4 LPTIM1 interrupt clear register [alternate] (LPTIM1_ICR) 2464
46.7.5 LPTIM1 interrupt enable register [alternate] (LPTIM1_DIER) 2466
46.7.6 LPTIM1 interrupt enable register [alternate] (LPTIM1_DIER) 2467
46.7.7 LPTIM configuration register (LPTIM_CFGR) 2469

46.7.8 LPTIM control register (LPTIM_CR) 2471
46.7.9 LPTIM compare register 1 (LPTIM_CCR1) 2472
46.7.10 LPTIM autoreload register (LPTIM_ARR) 2473
46.7.11 LPTIM counter register (LPTIM_CNT) 2473
46.7.12 LPTIM configuration register 2 (LPTIM_CFGR2) 2474
46.7.13 LPTIM repetition register (LPTIM_RCR) 2475
46.7.14 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) 2476
46.7.15 LPTIM compare register 2 (LPTIM_CCR2) 2478
46.7.16 LPTIM register map 2479

47 Independent watchdog (IWDG) 2481

47.1 IWDG introduction 2481
47.2 IWDG main features 2481
47.3 IWDG implementation 2481
47.4 IWDG functional description 2482
47.4.1 IWDG block diagram 2482
47.4.2 IWDG internal signals 2483
47.4.3 Software and hardware watchdog modes 2483
47.4.4 Window option 2484
47.4.5 Debug 2487
47.4.6 Register access protection 2487
47.5 IWDG low power modes 2488
47.6 IWDG interrupts 2488
47.7 IWDG registers 2490
47.7.1 IWDG key register (IWDG_KR) 2491
47.7.2 IWDG prescaler register (IWDG_PR) 2491
47.7.3 IWDG reload register (IWDG_RLR) 2492
47.7.4 IWDG status register (IWDG_SR) 2492
47.7.5 IWDG window register (IWDG_WINR) 2494
47.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 2494
47.7.7 IWDG interrupt clear register (IWDG_ICR) 2495
47.7.8 IWDG register map 2496

48 System window watchdog (WWDG) 2497

48.1 WWDG introduction 2497
48.2 WWDG main features 2497

48.3 WWDG implementation 2497
48.4 WWDG functional description 2498
48.4.1 WWDG block diagram 2498
48.4.2 WWDG internal signals 2498
48.4.3 Enabling the watchdog 2499
48.4.4 Controlling the down-counter 2499
48.4.5 How to program the watchdog timeout 2499
48.4.6 Debug mode 2500
48.5 WWDG interrupts 2501
48.6 WWDG registers 2501
48.6.1 WWDG control register (WWDG_CR) 2501
48.6.2 WWDG configuration register (WWDG_CFR) 2502
48.6.3 WWDG status register (WWDG_SR) 2503
48.6.4 WWDG register map 2503

49 Real-time clock (RTC) 2504
49.1 RTC introduction 2504
49.2 RTC main features 2504
49.3 RTC functional description 2504
49.3.1 RTC block diagram 2504
49.3.2 RTC pins and internal signals 2506
49.3.3 GPIOs controlled by the RTC 2507
49.3.4 RTC registers write protection 2507
49.3.5 RTC secure protection modes 2507
49.3.6 RTC privilege protection modes 2509
49.3.7 Clock and prescalers 2510
49.3.8 Real-time clock and calendar 2511
49.3.9 Calendar ultra-low power mode 2512
49.3.10 Programmable alarms 2512
49.3.11 Periodic auto-wake-up 2512
49.3.12 RTC initialization and configuration 2513
49.3.13 Reading the calendar 2516
49.3.14 Resetting the RTC 2517
49.3.15 RTC synchronization 2517
49.3.16 RTC reference clock detection 2518
49.3.17 RTC smooth digital calibration 2519

49.3.18 Timestamp function 2521
49.3.19 Calibration clock output 2522
49.3.20 Tamper and alarm output 2522
49.4 RTC low-power modes 2523
49.5 RTC interrupts 2523
49.6 RTC registers 2525
49.6.1 RTC time register (RTC_TR) 2525
49.6.2 RTC date register (RTC_DR) 2526
49.6.3 RTC subsecond register (RTC_SSR) 2527
49.6.4 RTC initialization control and status register (RTC_ICSR) 2528
49.6.5 RTC prescaler register (RTC_PRER) 2530
49.6.6 RTC wake-up timer register (RTC_WUTR) 2531
49.6.7 RTC control register (RTC_CR) 2532
49.6.8 RTC privilege mode control register (RTC_PRIVCFGR) 2536
49.6.9 RTC secure configuration register (RTC_SECCFGR) 2537
49.6.10 RTC write protection register (RTC_WPR) 2538
49.6.11 RTC calibration register (RTC_CALR) 2539
49.6.12 RTC shift control register (RTC_SHIFTR) 2540
49.6.13 RTC timestamp time register (RTC_TSTR) 2541
49.6.14 RTC timestamp date register (RTC_TSDR) 2542
49.6.15 RTC timestamp subsecond register (RTC_TSSSR) 2543
49.6.16 RTC alarm A register (RTC_ALRMAR) 2543
49.6.17 RTC alarm A subsecond register (RTC_ALRMASSR) 2545
49.6.18 RTC alarm B register (RTC_ALRMBR) 2546
49.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR) 2547
49.6.20 RTC status register (RTC_SR) 2548
49.6.21 RTC nonsecure masked interrupt status register (RTC_MISR) 2549
49.6.22 RTC secure masked interrupt status register (RTC_SMISR) 2550
49.6.23 RTC status clear register (RTC_SCR) 2551
49.6.24 RTC timestamp on tamper control register (RTC_TAMPTSCR) 2552
49.6.25 RTC timestamp status register (RTC_TSIDR) 2553
49.6.26 RTC alarm A binary mode register (RTC_ALRABINR) 2554
49.6.27 RTC alarm B binary mode register (RTC_ALRBBINR) 2554
49.6.28 RTC register map 2555

50 Tamper and backup registers (TAMP) 2558

50.1 TAMP introduction 2558

50.2 TAMP main features . . . 2558
50.3 TAMP functional description . . . 2559
50.3.1 TAMP block diagram . . . 2559
50.3.2 TAMP pins and internal signals . . . 2560
50.3.3 GPIOs controlled by the RTC and TAMP . . . 2563
50.3.4 TAMP register write protection . . . 2565
50.3.5 TAMP secure protection modes . . . 2565
50.3.6 Backup register protection zones . . . 2566
50.3.7 TAMP privilege protection modes . . . 2566
50.3.8 Boot hardware key (BHK) . . . 2567
50.3.9 Tamper detection . . . 2567
50.3.10 TAMP backup registers and other device secret erase . . . 2568
50.3.11 Tamper detection configuration and initialization . . . 2570
50.4 TAMP low-power modes . . . 2576
50.5 TAMP interrupts . . . 2577
50.6 TAMP registers . . . 2577
50.6.1 TAMP control register 1 (TAMP_CR1) . . . 2577
50.6.2 TAMP control register 2 (TAMP_CR2) . . . 2579
50.6.3 TAMP control register 3 (TAMP_CR3) . . . 2581
50.6.4 TAMP filter control register (TAMP_FLTCR) . . . 2582
50.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) . . . 2583
50.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) . . . 2585
50.6.7 TAMP active tamper output register (TAMP_ATOR) . . . 2585
50.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) . . . 2586
50.6.9 TAMP secure configuration register (TAMP_SECCFGR) . . . 2587
50.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) . . . 2588
50.6.11 TAMP interrupt enable register (TAMP_IER) . . . 2589
50.6.12 TAMP status register (TAMP_SR) . . . 2591
50.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR) . . . 2593
50.6.14 TAMP secure masked interrupt status register (TAMP_SMISR) . . . 2595
50.6.15 TAMP status clear register (TAMP_SCR) . . . 2596
50.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R) . . . 2598
50.6.17 TAMP option register (TAMP_OR) . . . 2598
50.6.18 TAMP resources protection configuration register (TAMP_RPCFGR) 2599
50.6.19 TAMP backup x register (TAMP_BKPxR) . . . 2601
50.6.20 TAMP register map . . . 2602

51 Inter-integrated circuit interface (I2C) 2604

51.1 I2C introduction 2604

51.2 I2C main features 2604

51.3 I2C implementation 2605

51.4 I2C functional description 2605

51.4.1 I2C block diagram 2606

51.4.2 I2C pins and internal signals 2606

51.4.3 I2C clock requirements 2607

51.4.4 I2C mode selection 2607

51.4.5 I2C initialization 2608

51.4.6 I2C reset 2612

51.4.7 I2C data transfer 2613

51.4.8 I2C target mode 2615

51.4.9 I2C controller mode 2624

51.4.10 I2C_TIMINGR register configuration examples 2635

51.4.11 SMBus specific features 2637

51.4.12 SMBus initialization 2639

51.4.13 SMBus I2C_TIMEOUTR register configuration examples 2641

51.4.14 SMBus target mode 2642

51.4.15 SMBus controller mode 2645

51.4.16 Wake-up from Stop mode on address match 2648

51.4.17 Error conditions 2649

51.5 I2C in low-power modes 2651

51.6 I2C interrupts 2651

51.7 I2C DMA requests 2652

51.7.1 Transmission using DMA 2652

51.7.2 Reception using DMA 2652

51.8 I2C debug modes 2652

51.9 I2C registers 2653

51.9.1 I2C control register 1 (I2C_CR1) 2653

51.9.2 I2C control register 2 (I2C_CR2) 2656

51.9.3 I2C own address 1 register (I2C_OAR1) 2658

51.9.4 I2C own address 2 register (I2C_OAR2) 2658

51.9.5 I2C timing register (I2C_TIMINGR) 2659

51.9.6 I2C timeout register (I2C_TIMEOUTR) 2660

51.9.7 I2C interrupt and status register (I2C_ISR) 2661

51.9.8 I2C interrupt clear register (I2C_ICR) 2664
51.9.9 I2C PEC register (I2C_PECR) 2665
51.9.10 I2C receive data register (I2C_RXDR) 2665
51.9.11 I2C transmit data register (I2C_TXDR) 2666
51.9.12 I2C register map 2667

52 Improved inter-integrated circuit (I3C) 2668

52.1 I3C introduction 2668
52.2 I3C main features 2668
52.2.1 I3C MIPI® support 2670
52.2.2 I3C without clock stretching by the target 2671
52.3 I3C implementation 2673
52.3.1 I3C instantiation 2673
52.3.2 I3C wake-up from low-power mode(s) 2673
52.3.3 I3C FIFOs 2673
52.3.4 I3C triggers 2673
52.3.5 I3C interrupt(s) 2673
52.4 I3C block diagram 2674
52.5 I3C pins and internal signals 2674
52.6 I3C reset and clocks 2675
52.6.1 I3C reset 2675
52.6.2 I3C clocks and requirements 2675
52.7 I3C peripheral state and programming 2677
52.7.1 I3C peripheral state 2677
52.7.2 I3C controller state and programming sequence 2678
52.7.3 I3C target state and programming sequence 2683
52.8 I3C registers and programming 2687
52.8.1 I3C register set, as controller/target 2687
52.8.2 I3C registers and fields use versus peripheral state, as controller . . 2688
52.8.3 I3C registers and fields usage versus peripheral state, as target . . . 2691
52.9 I3C bus transfers and programming 2693
52.9.1 I3C command set (CCCs), as controller/target 2693
52.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller 2697
52.9.3 I3C broadcast ENTDAA CCC transfer, as controller 2699
52.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller 2699

52.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target 2701
52.9.6 I3C broadcast ENTDAA CCC transfer, as target 2703
52.9.7 I3C broadcast DEFTGTS CCC transfer, as target 2704
52.9.8 I3C broadcast DEFGRPA CCC transfer, as target 2705
52.9.9 I3C direct GETSTATUS CCC response, as target 2706
52.9.10 I3C private read/write transfer, as controller 2707
52.9.11 I3C private read/write transfer, as target 2708
52.9.12 Legacy I2C read/write transfer, as controller 2709
52.9.13 I3C IBI transfer, as controller/target 2710
52.9.14 I3C hot-join request transfer, as controller/target 2712
52.9.15 I3C controller-role request transfer, as controller/target 2713
52.10 I3C FIFOs management, as controller 2714
52.10.1 C-FIFO management, as controller 2714
52.10.2 TX-FIFO management, as controller 2715
52.10.3 RX-FIFO management, as controller 2718
52.10.4 S-FIFO management, as controller 2720
52.11 I3C FIFOs management, as target 2722
52.11.1 RX-FIFO management, as target 2722
52.11.2 TX-FIFO management, as target 2723
52.12 I3C error management 2726
52.12.1 Controller error management 2726
52.12.2 Target error management 2728
52.13 I3C wake-up from low-power mode(s) 2729
52.13.1 Wake-up from Stop 2729
52.14 I3C in low-power modes 2732
52.15 I3C interrupts 2733
52.16 I3C registers 2733
52.16.1 I3C message control register (I3C_CR) 2733
52.16.2 I3C message control register [alternate] (I3C_CR) 2735
52.16.3 I3C configuration register (I3C_CFGR) 2737
52.16.4 I3C receive data byte register (I3C_RDR) 2742
52.16.5 I3C receive data word register (I3C_RDWR) 2743
52.16.6 I3C transmit data byte register (I3C_TDR) 2744
52.16.7 I3C transmit data word register (I3C_TDWR) 2745
52.16.8 I3C IBI payload data register (I3C_IBIDR) 2746

52.16.9 I3C target transmit configuration register (I3C_TGTTDR) 2747
52.16.10 I3C status register (I3C_SR) 2748
52.16.11 I3C status error register (I3C_SER) 2749
52.16.12 I3C received message register (I3C_RMR) 2751
52.16.13 I3C event register (I3C_EVR) 2752
52.16.14 I3C interrupt enable register (I3C_IER) 2756
52.16.15 I3C clear event register (I3C_CEVR) 2758
52.16.16 I3C masked interrupt status register (I3C_MISR) 2760
52.16.17 I3C own device characteristics register (I3C_DEVR0) 2761
52.16.18 I3C device x characteristics register (I3C_DEVRx) 2763
52.16.19 I3C maximum read length register (I3C_MAXRLR) 2765
52.16.20 I3C maximum write length register (I3C_MAXWLR) 2766
52.16.21 I3C timing register 0 (I3C_TIMINGR0) 2767
52.16.22 I3C timing register 1 (I3C_TIMINGR1) 2768
52.16.23 I3C timing register 2 (I3C_TIMINGR2) 2770
52.16.24 I3C bus characteristics register (I3C_BCR) 2771
52.16.25 I3C device characteristics register (I3C_DCR) 2772
52.16.26 I3C get capability register (I3C_GETCAPR) 2773
52.16.27 I3C controller-role capability register (I3C_CRCAPR) 2774
52.16.28 I3C get max data speed register (I3C_GETMXDSR) 2775
52.16.29 I3C extended provisioned ID register (I3C_EPIDR) 2777
52.16.30 I3C register map 2778

53 Universal synchronous/asynchronous receiver transmitter (USART/UART) 2781

53.1 USART introduction 2781
53.2 USART main features 2781
53.3 USART extended features 2782
53.4 USART implementation 2782
53.5 USART functional description 2784
53.5.1 USART block diagram 2784
53.5.2 USART pins and internal signals 2784
53.5.3 USART clocks 2786
53.5.4 USART character description 2786
53.5.5 USART FIFOs and thresholds 2789
53.5.6 USART transmitter 2789
53.5.7 USART receiver 2792

53.5.8 USART baud rate generation . . . 2799
53.5.9 Tolerance of the USART receiver to clock deviation . . . 2801
53.5.10 USART auto baud rate detection . . . 2802
53.5.11 USART multiprocessor communication . . . 2804
53.5.12 USART Modbus communication . . . 2806
53.5.13 USART parity control . . . 2807
53.5.14 USART LIN (local interconnection network) mode . . . 2808
53.5.15 USART synchronous mode . . . 2810
53.5.16 USART single-wire half-duplex communication . . . 2814
53.5.17 USART receiver timeout . . . 2814
53.5.18 USART smartcard mode . . . 2815
53.5.19 USART IrDA SIR ENDEC block . . . 2819
53.5.20 Continuous communication using USART and DMA . . . 2822
53.5.21 RS232 hardware flow control and RS485 driver enable . . . 2824
53.5.22 USART low-power management . . . 2827
53.6 USART in low-power modes . . . 2830
53.7 USART interrupts . . . 2830
53.8 USART registers . . . 2833
53.8.1 USART control register 1 (USART_CR1) . . . 2833
53.8.2 USART control register 1 [alternate] (USART_CR1) . . . 2837
53.8.3 USART control register 2 (USART_CR2) . . . 2840
53.8.4 USART control register 3 (USART_CR3) . . . 2844
53.8.5 USART control register 3 [alternate] (USART_CR3) . . . 2848
53.8.6 USART baud rate register (USART_BRR) . . . 2851
53.8.7 USART guard time and prescaler register (USART_GTPR) . . . 2852
53.8.8 USART receiver timeout register (USART_RTOR) . . . 2853
53.8.9 USART request register (USART_RQR) . . . 2854
53.8.10 USART interrupt and status register (USART_ISR) . . . 2855
53.8.11 USART interrupt and status register [alternate] (USART_ISR) . . . 2861
53.8.12 USART interrupt flag clear register (USART_ICR) . . . 2866
53.8.13 USART receive data register (USART_RDR) . . . 2867
53.8.14 USART transmit data register (USART_TDR) . . . 2868
53.8.15 USART prescaler register (USART_PRESC) . . . 2868
53.8.16 USART register map . . . 2869

54 Low-power universal asynchronous receiver transmitter (LPUART) . . . 2871

54.1 LPUART introduction 2871
54.2 LPUART main features 2871
54.3 LPUART implementation 2872
54.4 LPUART functional description 2873
54.4.1 LPUART block diagram 2873
54.4.2 LPUART pins and internal signals 2874
54.4.3 LPUART clocks 2875
54.4.4 LPUART character description 2875
54.4.5 LPUART FIFOs and thresholds 2877
54.4.6 LPUART transmitter 2877
54.4.7 LPUART receiver 2881
54.4.8 LPUART baud rate generation 2885
54.4.9 Tolerance of the LPUART receiver to clock deviation 2886
54.4.10 LPUART multiprocessor communication 2887
54.4.11 LPUART parity control 2889
54.4.12 LPUART single-wire half-duplex communication 2890
54.4.13 Continuous communication using DMA and LPUART 2890
54.4.14 RS232 hardware flow control and RS485 driver enable 2893
54.4.15 LPUART low-power management 2895
54.5 LPUART in low-power modes 2898
54.6 LPUART interrupts 2899
54.7 LPUART registers 2900
54.7.1 LPUART control register 1 (LPUART_CR1) 2900
54.7.2 LPUART control register 1 [alternate] (LPUART_CR1) 2903
54.7.3 LPUART control register 2 (LPUART_CR2) 2906
54.7.4 LPUART control register 3 (LPUART_CR3) 2908
54.7.5 LPUART control register 3 [alternate] (LPUART_CR3) 2911
54.7.6 LPUART baud rate register (LPUART_BRR) 2913
54.7.7 LPUART request register (LPUART_RQR) 2913
54.7.8 LPUART interrupt and status register (LPUART_ISR) 2914
54.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 2919
54.7.10 LPUART interrupt flag clear register (LPUART_ICR) 2922
54.7.11 LPUART receive data register (LPUART_RDR) 2923
54.7.12 LPUART transmit data register (LPUART_TDR) 2923
54.7.13 LPUART prescaler register (LPUART_PRESC) 2924
54.7.14 LPUART register map 2925

55 Serial peripheral interface (SPI) 2927

55.1 SPI introduction 2927

55.2 SPI main features 2927

55.3 SPI implementation 2928

55.4 SPI functional description 2929

55.4.1 SPI block diagram 2929

55.4.2 SPI pins and internal signals 2930

55.4.3 SPI communication general aspects 2931

55.4.4 Communications between one master and one slave 2931

55.4.5 Standard multislave communication 2934

55.4.6 Multimaster communication 2937

55.4.7 Slave select (NSS pin) management 2938

55.4.8 Ready pin (RDY) management 2942

55.4.9 Communication formats 2942

55.4.10 Configuring the SPI 2944

55.4.11 Enabling the SPI 2945

55.4.12 SPI data transmission and reception procedures 2946

55.4.13 Disabling the SPI 2950

55.4.14 Communication using DMA (direct memory addressing) 2951

55.5 SPI specific modes and control 2953

55.5.1 TI mode 2953

55.5.2 SPI error flags 2953

55.5.3 CRC computation 2957

55.6 SPI in low-power modes 2958

55.7 SPI interrupts 2958

55.8 I2S main features 2960

55.9 I2S functional description 2960

55.9.1 I2S general description 2960

55.9.2 Pin sharing with SPI function 2961

55.9.3 Bitfields usable in I2S/PCM mode 2961

55.9.4 Slave and master modes 2962

55.9.5 Supported audio protocols 2962

55.9.6 Additional serial interface flexibility 2968

55.9.7 Startup sequence 2970

55.9.8 Stop sequence 2972

55.9.9 Clock generator 2973

55.9.10 Internal FIFOs . . . 2975
55.9.11 FIFO status flags . . . 2976
55.9.12 Handling of underrun situation . . . 2976
55.9.13 Handling of overrun situation . . . 2977
55.9.14 Frame error detection . . . 2978
55.9.15 DMA interface . . . 2980
55.9.16 Programing examples . . . 2980
55.10 I2S interrupts . . . 2982
55.11 SPI/I2S registers . . . 2983
55.11.1 SPI/I2S control register 1 (SPI_CR1) . . . 2983
55.11.2 SPI/I2S control register 2 (SPI_CR2) . . . 2985
55.11.3 SPI/I2S configuration register 1 (SPI_CFG1) . . . 2985
55.11.4 SPI/I2S configuration register 2 (SPI_CFG2) . . . 2989
55.11.5 SPI/I2S interrupt enable register (SPI_IER) . . . 2991
55.11.6 SPI/I2S status register (SPI_SR) . . . 2992
55.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) . . . 2995
55.11.8 SPI/I2S transmit data register (SPI_TXDR) . . . 2996
55.11.9 SPI/I2S receive data register (SPI_RXDR) . . . 2996
55.11.10 SPI/I2S polynomial register (SPI_CRCPOLY) . . . 2997
55.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC) . . . 2997
55.11.12 SPI/I2S receiver CRC register (SPI_RXCRC) . . . 2998
55.11.13 SPI/I2S underrun data register (SPI_UDRDR) . . . 2999
55.11.14 SPI/I2S configuration register (SPI_I2SCFGR) . . . 2999
55.11.15 SPI/I2S register map . . . 3001

56 FD controller area network (FDCAN) . . . 3003
56.1 FDCAN introduction . . . 3003
56.2 FDCAN main features . . . 3005
56.3 FDCAN functional description . . . 3006
56.3.1 FDCAN block diagram . . . 3006
56.3.2 FDCAN pins and internal signals . . . 3007
56.3.3 Bit timing . . . 3008
56.3.4 Operating modes . . . 3009
56.3.5 Error management . . . 3018
56.3.6 Message RAM . . . 3019
56.3.7 FIFO acknowledge handling . . . 3028
56.3.8 FDCAN Rx FIFO element . . . 3028

56.3.9 FDCAN Tx buffer element 3030
56.3.10 FDCAN Tx event FIFO element 3032
56.3.11 FDCAN standard message ID filter element 3033
56.3.12 FDCAN extended message ID filter element 3034
56.4 FDCAN registers 3036
56.4.1 FDCAN core release register (FDCAN_CREL) 3036
56.4.2 FDCAN endian register (FDCAN_ENDN) 3036
56.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 3036
56.4.4 FDCAN test register (FDCAN_TEST) 3037
56.4.5 FDCAN RAM watchdog register (FDCAN_RWD) 3038
56.4.6 FDCAN CC control register (FDCAN_CCCR) 3039
56.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 3040
56.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 3042
56.4.9 FDCAN timestamp counter value register (FDCAN_TSCV) 3042
56.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 3043
56.4.11 FDCAN timeout counter value register (FDCAN_TOCV) 3044
56.4.12 FDCAN error counter register (FDCAN_ECR) 3044
56.4.13 FDCAN protocol status register (FDCAN_PSR) 3045
56.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 3047
56.4.15 FDCAN interrupt register (FDCAN_IR) 3047
56.4.16 FDCAN interrupt enable register (FDCAN_IE) 3050
56.4.17 FDCAN interrupt line select register (FDCAN_ILS) 3052
56.4.18 FDCAN interrupt line enable register (FDCAN_ILE) 3053
56.4.19 FDCAN global filter configuration register (FDCAN_RXGFC) 3053
56.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM) 3055
56.4.21 FDCAN high-priority message status register (FDCAN_HPMS) 3055
56.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 3056
56.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 3057
56.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 3057
56.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 3058
56.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC) 3058
56.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 3059
56.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 3059
56.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR) 3060
56.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 3061
56.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 3061
56.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 3062

56.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) . . . 3062
56.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) . . . 3063
56.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS) . . . 3063
56.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 3064
56.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV) . . . 3064
56.4.38 FDCAN register map . . . 3065

57 Position encoder interface (POSENC) . . . 3069

57.1 POSENC introduction . . . 3069
57.2 POSENC main features . . . 3069
57.3 POSENC functional description . . . 3070
57.3.1 POSENC block diagram . . . 3070
57.3.2 POSENC pins and internal signals . . . 3070
57.3.3 POSENC reset and clocks . . . 3072
57.3.4 POSENC memory organization and accesses . . . 3073
57.3.5 EnDat3 memory accesses . . . 3073
57.3.6 Hiperface DSL memory accesses . . . 3074
57.3.7 POSENC strobe interface . . . 3084
57.4 POSENC interrupts . . . 3085
57.5 POSENC initialization . . . 3092
57.6 POSENC registers . . . 3092
57.6.1 POSENC control register 1 (POSENC_CR1) . . . 3092
57.6.2 POSENC control register 2 (POSENC_CR2) . . . 3093
57.6.3 POSENC interrupt enable register (POSENC_IER) . . . 3094
57.6.4 POSENC interrupt clear register (POSENC_ICR) . . . 3095
57.6.5 POSENC masked interrupt status register (POSENC_MISR) . . . 3096
57.6.6 POSENC interrupt status register (POSENC_ISR) . . . 3098
57.6.7 POSENC interrupt status register 1 (POSENC_SR1) . . . 3099
57.6.8 POSENC interrupt status register 2 (POSENC_SR2) . . . 3100
57.6.9 POSENC interrupt status register 3 (POSENC_SR3) . . . 3100
57.6.10 POSENC data 0 address register (POSENC_D0AR) . . . 3101
57.6.11 POSENC data 1 address register (POSENC_D1AR) . . . 3101
57.6.12 POSENC data 0 register (POSENC_D0R) . . . 3102
57.6.13 POSENC data 1 register (POSENC_D1R) . . . 3102

57.6.14 POSENC Hiperface-DSL safe position 1 CRC register (POSENC_HD_VPOSCRCR) 3102
57.6.15 POSENC Hiperface-DSL safe position 1 VPOS4 register (POSENC_HD_VPOS4R) 3103
57.6.16 POSENC Hiperface-DSL safe position 1 VPOS3_0 register (POSENC_HD_VPOS30R) 3103
57.6.17 POSENC Hiperface-DSL safe position 1 VEL2_0 register (POSENC_HD_VEL20R) 3104
57.6.18 POSENC Hiperface-DSL fast position POS4 register (POSENC_HD_POS4R) 3104
57.6.19 POSENC Hiperface-DSL fast position POS3_0 register (POSENC_HD_POS30R) 3105
57.6.20 POSENC register map 3106

58 Universal serial bus full-speed host/device interface (USB) 3108

58.1 USB introduction 3108
58.2 USB main features 3108
58.3 USB implementation 3108
58.4 USB functional description 3109
58.4.1 USB block diagram 3109
58.4.2 USB pins and internal signals 3110
58.4.3 USB reset and clocks 3110
58.4.4 General description and Device mode functionality 3110
58.4.5 Description of USB blocks used in both Device and Host modes . . . 3111
58.4.6 Description of host frame scheduler (HFS) specific to Host mode . . 3112
58.5 Programming considerations for Device and Host modes 3113
58.5.1 Generic USB Device programming 3113
58.5.2 System and power-on reset 3114
58.5.3 Double-buffered endpoints and usage in Device mode 3121
58.5.4 Double buffered channels: usage in Host mode 3123
58.5.5 Isochronous transfers in Device mode 3124
58.5.6 Isochronous transfers in Host mode 3125
58.5.7 Suspend/resume events 3126
58.6 USB registers 3129
58.6.1 USB control register (USB_CNTR) 3130
58.6.2 USB interrupt status register (USB_ISTR) 3133
58.6.3 USB frame number register (USB_FNR) 3137
58.6.4 USB Device address (USB_DADDR) 3137

58.6.5 USB LPM control and status register (USB_LPMCSR) 3138
58.6.6 USB battery charging detector (USB_BCDR) 3139
58.6.7 USB endpoint/channel n register (USB_CHEPnR) 3140
58.7 USBSRAM registers 3149
58.7.1 Channel/endpoint transmit buffer descriptor n (USBSRAM_CHEP_TXRXBD_n) 3150
58.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USBSRAM_CHEP_TXRXBD_n) 3150
58.7.3 Channel/endpoint receive buffer descriptor n (USBSRAM_CHEP_RXTXBD_n) 3152
58.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USBSRAM_CHEP_RXTXBD_n) 3153
58.7.5 USBSRAM register map 3154

59 USB Type-C®/USB Power Delivery interface (UCPD) 3155
59.1 UCPD introduction 3155
59.2 UCPD main features 3155
59.3 UCPD implementation 3156
59.4 UCPD functional description 3156
59.4.1 UCPD block diagram 3157
59.4.2 UCPD reset and clocks 3158
59.4.3 Physical layer protocol 3159
59.4.4 UCPD BMC transmitter 3165
59.4.5 UCPD BMC receiver 3167
59.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd) 3168
59.4.7 UCPD Type-C voltage monitoring and de-bouncing 3169
59.4.8 UCPD fast role swap (FRS) 3169
59.4.9 UCPD DMA Interface 3169
59.4.10 Wake-up from Stop mode 3169
59.5 UCPD programming sequences 3170
59.5.1 Initialization phase 3170
59.5.2 Type-C state machine handling 3170
59.5.3 USB PD transmit 3172
59.5.4 USB PD receive 3173
59.5.5 UCPD software trimming 3174
59.6 UCPD low-power modes 3174
59.7 UCPD interrupts 3175

59.8 UCPD registers . . . 3176
59.8.1 UCPD configuration register 1 (UCPD_CFGR1) . . . 3176
59.8.2 UCPD configuration register 2 (UCPD_CFGR2) . . . 3178
59.8.3 UCPD configuration register 3 (UCPD_CFGR3) . . . 3178
59.8.4 UCPD control register (UCPD_CR) . . . 3179
59.8.5 UCPD interrupt mask register (UCPD_IMR) . . . 3182
59.8.6 UCPD status register (UCPD_SR) . . . 3183
59.8.7 UCPD interrupt clear register (UCPD_ICR) . . . 3186
59.8.8 UCPD Tx ordered set type register (UCPD_TX_ORDSETR) . . . 3187
59.8.9 UCPD Tx payload size register (UCPD_TX_PAYSZR) . . . 3187
59.8.10 UCPD Tx data register (UCPD_TXDR) . . . 3188
59.8.11 UCPD Rx ordered set register (UCPD_RX_ORDSETR) . . . 3188
59.8.12 UCPD Rx payload size register (UCPD_RX_PAYSZR) . . . 3189
59.8.13 UCPD receive data register (UCPD_RXDR) . . . 3190
59.8.14 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1) . . . 3190
59.8.15 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2) . . . 3191
59.8.16 UCPD register map . . . 3191

60 Debug support (DBG) . . . 3194
60.1 Introduction . . . 3194
60.2 DBG functional description . . . 3195
60.2.1 DBG block diagram . . . 3195
60.2.2 DBG pins and internal signals . . . 3195
60.2.3 DBG reset and clocks . . . 3196
60.2.4 DBG power domains . . . 3196
60.2.5 Debug and low-power modes . . . 3196
60.2.6 Security . . . 3197
60.2.7 Debug authentication . . . 3199
60.3 Serial-wire and JTAG debug port (SWJ-DP) . . . 3201
60.3.1 JTAG debug port . . . 3203
60.3.2 Serial-wire debug port . . . 3205
60.3.3 Debug port registers . . . 3207
60.3.4 Debug port register map and reset values . . . 3213
60.4 Access ports . . . 3214
60.4.1 Access port registers . . . 3215

60.4.2 Access port register map . . . 3221
60.5 ROM tables . . . 3222
60.5.1 System ROM table registers . . . 3225
60.5.2 System ROM table register map . . . 3229
60.5.3 MCU ROM table registers . . . 3230
60.5.4 MCU ROM table register map . . . 3234
60.5.5 Processor ROM table registers . . . 3235
60.5.6 Processor ROM table register map . . . 3239
60.6 Data watchpoint and trace unit (DWT) . . . 3240
60.6.1 DWT registers . . . 3241
60.6.2 DWT register map . . . 3255
60.7 Instrumentation trace macrocell (ITM) . . . 3258
60.7.1 ITM registers . . . 3258
60.7.2 ITM register map . . . 3266
60.8 Breakpoint unit (BPU) . . . 3268
60.8.1 BPU registers . . . 3268
60.8.2 BPU register map . . . 3274
60.9 Embedded Trace Macrocell (ETM) . . . 3275
60.9.1 ETM registers . . . 3275
60.9.2 ETM register map . . . 3299
60.10 Trace port interface unit (TPIU) . . . 3303
60.10.1 TPIU registers . . . 3304
60.10.2 TPIU register map . . . 3313
60.11 Cross-trigger interface (CTI) . . . 3315
60.11.1 CTI registers . . . 3316
60.11.2 CTI register map . . . 3326
60.12 Microcontroller debug unit (DBGMCU) . . . 3328
60.12.1 Device ID . . . 3328
60.12.2 Low-power mode emulation . . . 3328
60.12.3 Peripheral clock freeze . . . 3329
60.12.4 DBGMCU registers . . . 3330
60.12.5 DBGMCU register map . . . 3345
60.13 References . . . 3348

61 Device electronic signature . . . 3349

61.1 Unique device ID register (96 bits) . . . 3349

61.2 Flash size data register 3350
61.3 Package data register 3350
61.4 Temperature calibration value 3351
61.5 Temperature calibration value low 3352
61.6 Temperature calibration value high 3352
61.7 PUF reference value (256bit) 3352

62 Important security notice 3354

63 Revision history 3355

List of tables

Table 1. Implementation of masters . . . 113
Table 2. Implementation of slaves . . . 114
Table 3. Securable peripherals by TZSC . . . 118
Table 4. TrustZone-aware peripherals . . . 120
Table 5. Memory map and peripheral register addresses . . . 125
Table 6. Configuring security attributes with IDAU and SAU . . . 138
Table 7. MPCWMx resources for STM32G55x/56x . . . 140
Table 8. MPCBBx resources for STM32G53x/54x . . . 140
Table 9. MPCBBx resources for STM32G55x/56x . . . 140
Table 10. DMA channel use (security) . . . 143
Table 11. Secure alternate function between peripherals and allocated I/Os . . . 146
Table 12. Nonsecure peripheral functions that cannot be connected to secure I/Os . . . 146
Table 13. Nonsecure peripheral functions that can be connected to secure I/Os . . . 147
Table 14. TrustZone-aware DBGMCU nonsecure accesses management . . . 148
Table 15. DMA channel use (privilege) . . . 151
Table 16. Internal tampers in TAMP . . . 155
Table 17. Effect of low-power modes on TAMP . . . 156
Table 18. Main product life-cycle transitions . . . 158
Table 19. Typical product life-cycle phases . . . 159
Table 20. Software intellectual property protection with PRODUCT_STAT . . . 162
Table 21. Boot mode when TrustZone is disabled (TZEN = 0xC3) . . . 166
Table 22. Boot mode when TrustZone is enabled (TZEN = 0xB4) for STM32G53x/54x/55x . . . 166
Table 23. Boot mode when TrustZone is enabled (TZEN = 0xB4) for STM32G56x . . . 166
Table 24. GTZC features . . . 170
Table 25. GTZC1 subblock address offset . . . 171
Table 26. MPCWM resource assignment . . . 171
Table 27. MPCBB resource assignment for STM32G53x/54x . . . 171
Table 28. MPCBB resource assignment for STM32G55x/56x . . . 171
Table 29. Secure properties of subregions A and B . . . 174
Table 30. Privileged properties of subregions A and B . . . 174
Table 31. GTZC interrupt request . . . 176
Table 32. GTZC1 TZSC register map and reset values . . . 195
Table 33. GTZC1 TZIC register map and reset values . . . 226
Table 34. GTZC1 MPCBBz register map and reset values (z = 1 to 3) . . . 230
Table 35. Internal SRAMs features . . . 232
Table 36. Effect of low-power modes on RAMCFG . . . 233
Table 37. RAMCFG interrupt requests . . . 234
Table 38. RAMCFG register map and reset values . . . 239
Table 39. Recommended number of wait states and programming delay . . . 248
Table 40. Flash memory OTP organization . . . 259
Table 41. Read-only public data organization . . . 260
Table 42. Memory map and swapping options (64 Kbytes) . . . 265
Table 43. Memory map and swapping options (128 Kbytes) . . . 266
Table 44. Memory map and swapping options (256 Kbytes) . . . 266
Table 45. Memory map and swapping options (512 Kbytes) . . . 267
Table 46. Recommended reactions to FLASH_OPSR contents . . . 271
Table 47. Option-byte organization . . . 274
Table 48. Specific OB modifying rules . . . 278

Table 49. OB modifiable in closed product . . . 278
Table 50. Option-byte key area . . . 280
Table 51. Default secure watermark . . . 287
Table 52. Flash memory TZ protection summary . . . 288
Table 53. TZ protection and bank or mass erase summary . . . 288
Table 54. Secure watermark-based area . . . 289
Table 55. Secure hide protection . . . 290
Table 56. HDP protections summary . . . 291
Table 57. Secure configuration block-based registers access conditions . . . 292
Table 58. Privilege protection summary . . . 292
Table 59. Privilege and mass or bank erase . . . 293
Table 60. Privilege configuration register access conditions (TZ enabled) . . . 293
Table 61. Privilege configuration register access conditions (TZ disabled) . . . 293
Table 62. Flash register accesses . . . 294
Table 63. Flash interface register protection summary . . . 296
Table 64. High-cycle area protection summary: access to data area address range . . . 297
Table 65. HDP protected definition . . . 297
Table 66. Privilege pages and the data area - access to data area address range . . . 297
Table 67. Product states, debug states, and debug policy . . . 298
Table 68. PRODUCT_STATE transitions . . . 300
Table 69. TZ OBK protection summary . . . 301
Table 70. OBK protection summary with TZ disabled . . . 301
Table 71. Access conditions to secure control register . . . 302
Table 72. Access conditions to nonsecure control register . . . 302
Table 73. OTP/RO access constraints . . . 303
Table 74. Macros for RSS services . . . 303
Table 75. RSS lib interface functions . . . 304
Table 76. NSS lib interface functions . . . 309
Table 77. Effect of low-power modes on the embedded flash memory . . . 310
Table 78. Locating ECC failure . . . 318
Table 79. Flash interrupt request . . . 321
Table 80. Register map and reset value table . . . 373
Table 81. ICACHE features for STM32G55x/56x . . . 379
Table 82. ICACHE features for STM32G53x/54x . . . 379
Table 83. TAG memory dimensioning parameters for n-way set associative operating mode (default) . . . 381
Table 84. TAG memory dimensioning parameters for direct-mapped cache mode . . . 383
Table 85. ICACHE cacheability for AHB transaction . . . 384
Table 86. Memory configurations . . . 385
Table 87. ICACHE remap region size, base address, and remap address . . . 386
Table 88. ICACHE read requests management (when ECC is enabled) . . . 389
Table 89. ICACHE interrupts . . . 391
Table 90. ICACHE register map and reset values . . . 397
Table 91. DCACHE features . . . 399
Table 92. TAG memory dimensioning parameters . . . 401
Table 93. DCACHE cacheability for AHB transaction . . . 403
Table 94. DCACHE accesses management (when ECC is enabled) . . . 409
Table 95. DCACHE interrupts . . . 412
Table 96. DCACHE register map and reset values . . . 419
Table 97. PWR input/output pins . . . 421
Table 98. PWR internal input/output signals . . . 421
Table 99. Low-power mode summary . . . 437

Table 100. Functionalities depending on the working mode . . . 437
Table 101. Sleep mode . . . 441
Table 102. Memory power-down block selection for STM32G53x/54x devices . . . 442
Table 103. Memory power-down block selection for STM32G55x/56x devices . . . 442
Table 104. Stop mode . . . 443
Table 105. I/O retention with pull-up and pull-down . . . 444
Table 106. Standby mode . . . 447
Table 107. Power modes output states versus MCU power modes . . . 447
Table 108. PWR security configuration summary . . . 448
Table 109. PWR interrupt requests . . . 450
Table 110. PWR register map and reset values . . . 464
Table 111. RCC input/output signals connected to package pins or balls . . . 468
Table 112. RCC internal input/output signals . . . 468
Table 113. Reset coverage summary . . . 474
Table 114. Reset source identification (RCC_RSR) . . . 475
Table 115. Oscillator states versus system modes . . . 478
Table 116. PLLs frequency limits . . . 492
Table 117. Resulting duty cycle examples . . . 494
Table 118. HSIKERON behavior . . . 498
Table 119. Peripheral clock distribution summary . . . 500
Table 120. HRTIM1 clock multiplication ratio . . . 512
Table 121. Clock gating rules for TIMs and HRTIM1 . . . 513
Table 122. POSENC1 and POSENC2 clock constraints . . . 515
Table 123. Clock gating rules for PLAY1 bus clock . . . 516
Table 124. Clock gating rules for PLAY1 kernel clock . . . 517
Table 125. PKA clock gating rules . . . 520
Table 126. RNG clock gating rules with CCB . . . 520
Table 127. RNG clock gating rules without CCB . . . 522
Table 128. Clock gating rules for generic peripherals . . . 534
Table 129. RCC security configuration summary . . . 537
Table 130. Clock protection summary . . . 540
Table 131. Interrupt sources and control . . . 543
Table 132. RCC register map and reset values . . . 645
Table 133. CRS features . . . 652
Table 134. CRS internal input/output signals . . . 653
Table 135. CRS interconnection . . . 654
Table 136. Effect of low-power modes on CRS . . . 657
Table 137. Interrupt control bits . . . 657
Table 138. CRS register map and reset values . . . 662
Table 139. Port bit configuration . . . 665
Table 140. GPIO secured bits . . . 673
Table 141. GPIO register map and reset values . . . 683
Table 142. SBS internal input/output signals . . . 687
Table 143. HDPL encoded values . . . 690
Table 144. SBS boot logic . . . 691
Table 145. OBK-HDPL logic . . . 696
Table 146. Bank swap control . . . 696
Table 147. SBS register map and reset values . . . 710
Table 148. Peripherals interconnect matrix . . . 714
Table 149. GPDMA1/2 channel implementation . . . 731
Table 150. GPDMA1/2 wake-up in low-power modes . . . 731
Table 151. Programmed GPDMA1/2 request . . . 731

Table 152. Programmed GPDMA1/2 request as a block request . . . 736
Table 153. GPDMA1/2 channel with peripheral early termination . . . 736
Table 154. Programmed GPDMA1/2 request with peripheral early termination . . . 736
Table 155. Programmed GPDMA1/2 trigger . . . 737
Table 156. Programmed GPDMA source/destination burst . . . 760
Table 157. Programmed data handling . . . 765
Table 158. Effect of low-power modes on GPDMA . . . 779
Table 159. GPDMA interrupt requests . . . 780
Table 160. GPDMA register map and reset values . . . 809
Table 161. GPDMA1/2 channel implementation . . . 814
Table 162. GPDMA1/2 wake-up in low-power modes . . . 814
Table 163. Programmed GPDMA1/2 request . . . 814
Table 164. Programmed GPDMA1/2 request as a block request . . . 819
Table 165. GPDMA1/2 channel with peripheral early termination . . . 819
Table 166. Programmed GPDMA1/2 request with peripheral early termination . . . 819
Table 167. Programmed GPDMA1/2 trigger . . . 820
Table 168. Programmed GPDMA source/destination burst . . . 843
Table 169. Programmed data handling . . . 848
Table 170. Effect of low-power modes on GPDMA . . . 862
Table 171. GPDMA interrupt requests . . . 863
Table 172. GPDMA register map and reset values . . . 892
Table 173. STM32G56x/55x/54x/53x vector table . . . 896
Table 174. EXTI signals . . . 904
Table 175. EVG signals . . . 905
Table 176. EXTI line connections . . . 905
Table 177. Masking functionality . . . 909
Table 178. Register protection overview . . . 910
Table 179. EXTI register map sections . . . 911
Table 180. EXTI register map and reset values . . . 936
Table 181. CRC internal input/output signals . . . 939
Table 182. CRC register map and reset values . . . 944
Table 183. CORDIC functions . . . 946
Table 184. Cosine parameters . . . 946
Table 185. Sine parameters . . . 947
Table 186. Phase parameters . . . 947
Table 187. Modulus parameters . . . 948
Table 188. Arctangent parameters . . . 949
Table 189. Hyperbolic cosine parameters . . . 949
Table 190. Hyperbolic sine parameters . . . 950
Table 191. Hyperbolic arctangent parameters . . . 950
Table 192. Natural logarithm parameters . . . 951
Table 193. Natural log scaling factors and corresponding ranges . . . 951
Table 194. Square root parameters . . . 952
Table 195. Square root scaling factors and corresponding ranges . . . 952
Table 196. Precision vs. number of iterations . . . 955
Table 197. CORDIC register map and reset value . . . 962
Table 198. Valid combinations for read and write methods . . . 976
Table 199. FMAC register map and reset values . . . 989
Table 200. PLAY pins . . . 992
Table 201. PLAY internal signals . . . 992
Table 202. PLAY1 logic array input n signal selection . . . 993
Table 203. PLAY1 output connections . . . 995

Table 204. Truth table example 1 . . . 1003
Table 205. Truth table example 2 . . . 1005
Table 206. Logic element n LUT input selection . . . 1007
Table 207. Logic element n register clock selection . . . 1008
Table 208. Output n source selection . . . 1011
Table 209. PLAY register map and reset values . . . 1028
Table 210. XSPI implementation . . . 1032
Table 211. XSPI input/output pins . . . 1034
Table 212. XSPI internal signals . . . 1034
Table 213. Command/address phase description . . . 1042
Table 214. OctaRAM command address bit assignment (based on 64 Mb OctaRAM) . . . 1053
Table 215. Address alignment cases . . . 1060
Table 216. XSPI interrupt requests . . . 1062
Table 217. XSPI register map and reset values . . . 1085
Table 218. STM32G56x/55x/54x/53x features . . . 1088
Table 219. DLYB internal input/output signals . . . 1089
Table 220. DLYB interconnection . . . 1089
Table 221. Delay block control . . . 1090
Table 222. DLYB register map and reset values . . . 1093
Table 223. ADC features . . . 1096
Table 224. ADC input/output pins . . . 1098
Table 225. ADC internal input/output signals . . . 1098
Table 226. ADC1/2/3 interconnection . . . 1098
Table 227. Configuring the trigger polarity for regular external triggers . . . 1123
Table 228. Configuring the trigger polarity for injected external triggers . . . 1124
Table 229. TSAR timings depending on resolution . . . 1134
Table 230. Offset computation versus data resolution . . . 1139
Table 231. 12-bit data formats . . . 1141
Table 232. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0) . . . 1142
Table 233. Analog window watchdog features . . . 1150
Table 234. Analog watchdog 1 channel selection . . . 1151
Table 235. Analog watchdog 1, 2, 3 comparison . . . 1152
Table 236. Maximum output results versus N and M . . . 1157
Table 237. Oversampler operating mode summary . . . 1161
Table 238. DELAY bits versus ADC resolution . . . 1169
Table 239. ADC interrupts . . . 1183
Table 240. ADC register map and reset values for each ADC . . . 1223
Table 241. ADC register map and reset values (master and slave ADC common registers) . . . 1227
Table 242. DTS internal input/output signals . . . 1229
Table 243. Sampling time configuration . . . 1232
Table 244. Trigger configuration . . . 1233
Table 245. Temperature sensor behavior in low-power modes . . . 1235
Table 246. DTS register map and reset values . . . 1243
Table 247. DAC features . . . 1246
Table 248. DAC input/output pins . . . 1248
Table 249. DAC internal input/output signals . . . 1248
Table 250. DAC interconnection . . . 1248
Table 251. Data format (case of 12-bit data) . . . 1252
Table 252. HFSEL description . . . 1252
Table 253. Sample and refresh timings . . . 1263

Table 254. Channel output modes summary . . . 1264
Table 255. Effect of low-power modes on DAC . . . 1267
Table 256. DAC interrupts . . . 1268
Table 257. DAC register map and reset values . . . 1278
Table 258. VREFBUF typical values . . . 1281
Table 259. VREF buffer modes . . . 1282
Table 260. VREFBUF register map and reset values . . . 1285
Table 261. COMPx_INP non-inverting input assignment (COMP1 to COMP8) . . . 1287
Table 262. COMPx_INM inverting input assignment (COMP1 to COMP8) . . . 1288
Table 263. COMPx blanking source assignment (COMP1 to COMP8) . . . 1288
Table 264. Comparator behavior in low-power modes . . . 1293
Table 265. Interrupt through EXTI control bits . . . 1293
Table 266. COMPx register map and reset values . . . 1296
Table 267. OPAMP pin and signal assignment per instance . . . 1298
Table 268. OPAMP pins . . . 1300
Table 269. OPAMP internal input/output signals . . . 1301
Table 270. Calibration loops . . . 1311
Table 271. Timer-controlled input selection . . . 1312
Table 272. Timer-controlled programmable gain selection . . . 1313
Table 273. Effect of low-power modes on OPAMP . . . 1313
Table 274. OPAMP register map and reset values . . . 1320
Table 275. Peripherals connected by CCB interconnect . . . 1322
Table 276. CCB internal input/output signals . . . 1323
Table 277. Blob creation steps for ECDSA signature (CCOP = 0b110000<i>0) . . . 1330
Table 278. Blob use steps for ECDSA signature (CCOP = 0b11000011) . . . 1333
Table 279. Blob use steps for signing public key computation (CCOP = 0b10000001) . . . 1335
Table 280. Blob creation steps for scalar multiplication (CCOP = 0b100000<i>0) . . . 1337
Table 281. Blob use steps for scalar multiplication (CCOP = 0b10000001) . . . 1338
Table 282. Blob creation steps for modular exponentiation (CCOP = 0b01000100) . . . 1339
Table 283. Blob use steps for modular exponentiation (CCOP = 0b01000101) . . . 1340
Table 284. CCB register map and reset values . . . 1346
Table 285. RNG internal input/output signals . . . 1348
Table 286. RNG interrupt requests . . . 1357
Table 287. RNG configurations . . . 1358
Table 288. Additional health test configurations . . . 1358
Table 289. Configuration selection . . . 1358
Table 290. RNG register map and reset map . . . 1366
Table 291. AES versus SAES features . . . 1368
Table 292. AES internal input/output signals . . . 1369
Table 293. AES approved symmetric key functions . . . 1370
Table 294. Counter mode initialization vector definition . . . 1379
Table 295. Initialization of IV registers in GCM mode . . . 1382
Table 296. GCM last block definition . . . 1382
Table 297. Initialization of IV registers in CCM mode . . . 1389
Table 298. AES data swapping example . . . 1392
Table 299. Key endianness in AES_KEYRx registers (128/256-bit keys) . . . 1394
Table 300. IVI bitfield spread over AES_IVRx registers . . . 1395
Table 301. AES interrupt requests . . . 1396
Table 302. Processing latency for ECB, CBC and CTR . . . 1397
Table 303. Processing latency for GCM and CCM (in clock cycles) . . . 1398
Table 304. AES register map and reset values . . . 1410
Table 305. AES versus SAES features . . . 1414

Table 306. SAES internal input/output signals . . . 1415
Table 307. SAES approved symmetric key functions . . . 1416
Table 308. Counter mode initialization vector definition . . . 1426
Table 309. Initialization of IV registers in GCM mode . . . 1429
Table 310. GCM last block definition . . . 1429
Table 311. Initialization of IV registers in CCM mode . . . 1435
Table 312. AES data swapping example . . . 1444
Table 313. Key endianness in SAES_KEYRx registers (128/256-bit keys) . . . 1446
Table 314. IVI bitfield spread over SAES_IVRx registers . . . 1448
Table 315. SAES interrupt requests . . . 1450
Table 316. Processing latency for ECB, CBC and CTR . . . 1451
Table 317. Processing latency for GCM and CCM (in SAES kernel clock cycles) . . . 1451
Table 318. SAES register map and reset values . . . 1466
Table 319. HASH internal input/output signals . . . 1469
Table 320. Information on supported hash algorithms . . . 1470
Table 321. Hash processor outputs . . . 1473
Table 322. Processing time (in clock cycle) . . . 1479
Table 323. HASH interrupt requests . . . 1480
Table 324. HASH1 register map and reset values . . . 1488
Table 325. Internal input/output signals . . . 1491
Table 326. PKA integer arithmetic functions list . . . 1492
Table 327. PKA prime field (Fp) elliptic curve functions list . . . 1492
Table 328. Example of ‘a’ curve coefficient for ECC Fp scalar . . . 1498
Table 329. Montgomery parameter computation . . . 1498
Table 330. Modular addition . . . 1499
Table 331. Modular subtraction . . . 1499
Table 332. Montgomery multiplication . . . 1500
Table 333. Modular exponentiation (normal mode) . . . 1501
Table 334. Modular exponentiation (fast mode) . . . 1501
Table 335. Modular inversion . . . 1502
Table 336. Modular reduction . . . 1502
Table 337. Arithmetic addition . . . 1502
Table 338. Arithmetic subtraction . . . 1503
Table 339. Arithmetic multiplication . . . 1503
Table 340. Arithmetic comparison . . . 1504
Table 341. CRT exponentiation . . . 1504
Table 342. Point on elliptic curve Fp check . . . 1505
Table 343. ECC Fp scalar multiplication . . . 1505
Table 344. ECDSA sign - Inputs . . . 1507
Table 345. ECDSA sign - Outputs . . . 1507
Table 346. Extended ECDSA sign - additional outputs . . . 1508
Table 347. ECDSA verification - inputs . . . 1508
Table 348. ECDSA verification - outputs . . . 1509
Table 349. ECC complete addition . . . 1509
Table 350. ECC double base ladder . . . 1510
Table 351. ECC projective to affine . . . 1511
Table 352. Family of supported curves for ECC operations . . . 1512
Table 353. Modular exponentiation . . . 1513
Table 354. ECC scalar multiplication . . . 1513
Table 355. ECDSA signature average computation time . . . 1514
Table 356. ECDSA verification average computation times . . . 1514
Table 357. ECC double base ladder average computation times . . . 1514

Table 358. ECC projective to affine average computation times . . . 1514
Table 359. ECC complete addition average computation times . . . 1514
Table 360. Point on elliptic curve Fp check average computation times . . . 1514
Table 361. Montgomery parameters average computation times . . . 1515
Table 362. PKA interrupt requests . . . 1515
Table 363. PKA register map and reset values . . . 1520
Table 364. Internal input/output signals . . . 1522
Table 365. PKA integer arithmetic functions list . . . 1523
Table 366. PKA prime field (Fp) elliptic curve functions list . . . 1524
Table 367. Chaining mode to/from PKA . . . 1529
Table 368. Integrity protected parameters list . . . 1531
Table 369. Confidentiality protected parameter list . . . 1531
Table 370. Example of ‘a’ curve coefficient for ECC Fp scalar . . . 1533
Table 371. Montgomery parameter computation . . . 1534
Table 372. Modular addition . . . 1534
Table 373. Modular subtraction . . . 1535
Table 374. Montgomery multiplication . . . 1536
Table 375. Modular exponentiation (normal mode) . . . 1537
Table 376. Modular exponentiation (fast mode) . . . 1537
Table 377. Modular exponentiation (protected mode) . . . 1538
Table 378. Modular inversion . . . 1538
Table 379. Modular reduction . . . 1539
Table 380. Arithmetic addition . . . 1539
Table 381. Arithmetic subtraction . . . 1539
Table 382. Arithmetic multiplication . . . 1540
Table 383. Arithmetic comparison . . . 1540
Table 384. CRT exponentiation . . . 1541
Table 385. Point on elliptic curve Fp check . . . 1542
Table 386. ECC Fp scalar multiplication . . . 1542
Table 387. ECDSA sign - Inputs . . . 1544
Table 388. ECDSA sign - Outputs . . . 1544
Table 389. Extended ECDSA sign - additional outputs . . . 1545
Table 390. ECDSA verification - inputs . . . 1545
Table 391. ECDSA verification - outputs . . . 1546
Table 392. ECC complete addition . . . 1546
Table 393. ECC double base ladder . . . 1547
Table 394. ECC projective to affine . . . 1548
Table 395. Family of supported curves for ECC operations . . . 1549
Table 396. Modular exponentiation . . . 1550
Table 397. ECC scalar multiplication . . . 1550
Table 398. ECDSA signature average computation time . . . 1551
Table 399. ECDSA verification average computation times . . . 1551
Table 400. ECC double base ladder average computation times . . . 1551
Table 401. ECC projective to affine average computation times . . . 1551
Table 402. ECC complete addition average computation times . . . 1551
Table 403. Point on elliptic curve Fp check average computation times . . . 1551
Table 404. Montgomery parameters average computation times . . . 1552
Table 405. PKA interrupt requests . . . 1552
Table 406. PKA register map and reset values . . . 1559
Table 407. OTFDEC internal input/output signals . . . 1561
Table 408. OTFDEC interrupt requests . . . 1565
Table 409. OTFDEC register map and reset values . . . 1579

Table 410. HRTIM inputs/outputs summary . . . 1587
Table 411. External events mapping and associated features . . . 1589
Table 412. Update enable inputs and sources . . . 1591
Table 413. Burst mode clock sources . . . 1591
Table 414. Burst mode trigger source . . . 1591
Table 415. Internal synchronization source . . . 1591
Table 416. Fault inputs . . . 1592
Table 417. External trigger inputs . . . 1592
Table 418. HRTIM DAC triggers connections . . . 1593
Table 419. Timer resolution and min. PWM frequency for f_{HRTIM} = 280 MHz . . . 1594
Table 420. Period and compare registers min and max values . . . 1598
Table 421. Timer operating modes . . . 1598
Table 422. Events mapping across timer A to F . . . 1604
Table 423. Interleaved mode selection . . . 1605
Table 424. Compare 1..3 values in interleaved mode . . . 1606
Table 425. Deadtime resolution and max absolute values . . . 1618
Table 426. Roll-over event destination and mode programming . . . 1634
Table 427. EExFLTR[4:0] codes depending on UDM bit setting . . . 1636
Table 428. External events features . . . 1642
Table 429. External events selection . . . 1642
Table 430. Output set/reset latency and jitter versus external event operating mode . . . 1644
Table 431. Filtering signals mapping per timer . . . 1647
Table 432. Windowing signals mapping per timer (EExFLTR[4:0] = 01111) . . . 1650
Table 433. HRTIM preloadable control registers and associated update sources . . . 1660
Table 434. Master timer update event propagation . . . 1665
Table 435. TIMx update event propagation . . . 1665
Table 436. Reset events able to generate an update . . . 1667
Table 437. Update event propagation for a timer reset . . . 1667
Table 438. Output state programming, x= A..F, y = 1 or 2 . . . 1669
Table 439. Timer output programming for burst mode . . . 1672
Table 440. Sampling rate and filter length vs FLTFxF[3:0] and clock setting . . . 1682
Table 441. Fault input blanking events . . . 1683
Table 442. Fault 1..6 counter reset source . . . 1684
Table 443. Effect of sync event versus timer operating modes . . . 1689
Table 444. DAC dual channel trigger example . . . 1697
Table 445. HRTIM interrupt summary . . . 1700
Table 446. HRTIM DMA request summary . . . 1702
Table 447. HRTIM Register map and reset values – master timer . . . 1878
Table 448. HRTIM register map and reset values – TIMx (x= A..F) . . . 1879
Table 449. HRTIM register map and reset values – common functions . . . 1884
Table 450. TIM input/output pins . . . 1891
Table 451. TIM internal input/output signals . . . 1891
Table 452. Interconnect to the tim_ti1 input multiplexer . . . 1893
Table 453. Interconnect to the tim_ti2 input multiplexer . . . 1893
Table 454. Interconnect to the tim_ti3 input multiplexer . . . 1893
Table 455. Interconnect to the tim_ti4 input multiplexer . . . 1893
Table 456. Internal trigger connection . . . 1894
Table 457. Interconnect to the tim_etr input multiplexer . . . 1895
Table 458. Interconnect to the tim_brk_cmp multiplexer . . . 1895
Table 459. Interconnect to the tim_brk2_cmp multiplexer . . . 1896
Table 460. Interconnect to tim_ocref_clr multiplexer . . . 1897
Table 461. System break interconnect . . . 1897

Table 462. CCR and ARR register change dithering pattern . . . 1931
Table 463. CCR register change dithering pattern in center-aligned PWM mode . . . 1932
Table 464. Asymmetric PWM modes . . . 1934
Table 465. Combined PWM mode overview . . . 1937
Table 466. Behavior of timer outputs versus tim_brk/tim_brk2 inputs . . . 1948
Table 467. Break protection disarming conditions . . . 1952
Table 468. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 1964
Table 469. Counting direction versus encoder signals and polarity settings . . . 1968
Table 470. DMA request . . . 1993
Table 471. Effect of low-power modes on TIM1/TIM8/TIM20 . . . 1994
Table 472. Interrupt requests . . . 1995
Table 473. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature . . . 2027
Table 474. TIM1/TIM8/TIM20 register map and reset values . . . 2065
Table 475. STM32G56x/55x/54x/53x general purpose timers . . . 2070
Table 476. TIM input/output pins . . . 2072
Table 477. TIM internal input/output signals . . . 2072
Table 478. Interconnect to the tim_ti1 input multiplexer . . . 2073
Table 479. Interconnect to the tim_ti2 input multiplexer . . . 2073
Table 480. Interconnect to the tim_ti3 input multiplexer . . . 2074
Table 481. Interconnect to the tim_ti4 input multiplexer . . . 2074
Table 482. TIMx internal trigger connection . . . 2075
Table 483. Interconnect to the tim_etr input multiplexer . . . 2075
Table 484. Interconnect to the tim_ocref_clr input multiplexer . . . 2076
Table 485. CCR and ARR register change dithering pattern . . . 2109
Table 486. CCR register change dithering pattern in center-aligned PWM mode . . . 2110
Table 487. Asymmetric PWM modes . . . 2111
Table 488. Combined PWM mode overview . . . 2112
Table 489. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 2123
Table 490. Counting direction versus encoder signals and polarity settings . . . 2128
Table 491. DMA request . . . 2157
Table 492. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5 . . . 2157
Table 493. Interrupt requests . . . 2158
Table 494. Output control bit for standard tim_ocx channels . . . 2182
Table 495. TIM2/TIM3/TIM4/TIM5 register map and reset values . . . 2194
Table 496. TIM internal input/output signals . . . 2198
Table 497. TIM_ARR register change dithering pattern . . . 2208
Table 498. DMA request . . . 2210
Table 499. Effect of low-power modes on TIM6/TIM7/TIM26/TIM27 . . . 2210
Table 500. Interrupt request . . . 2210
Table 501. TIM6/TIM7/TIM26/TIM27 register map and reset values . . . 2218
Table 502. TIM input/output pins . . . 2222
Table 503. TIM internal input/output signals . . . 2222
Table 504. Interconnect to the tim_ti1 input multiplexer . . . 2223
Table 505. Interconnect to the tim_ti2 input multiplexer . . . 2223
Table 506. TIMx internal trigger connection . . . 2224
Table 507. CCR and ARR register change dithering pattern . . . 2242
Table 508. Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 . . . 2254
Table 509. Interrupt requests . . . 2254
Table 510. Output control bit for standard tim_ocx channels . . . 2268
Table 511. TIM9/TIM12 register map and reset values . . . 2272
Table 512. Output control bit for standard tim_ocx channels . . . 2281

Table 513. TIM10/TIM11/TIM13/TIM14 register map and reset values . . . 2284
Table 514. TIM input/output pins . . . 2289
Table 515. TIM internal input/output signals . . . 2290
Table 516. Interconnect to the tim_ti1 input multiplexer . . . 2291
Table 517. Interconnect to the tim_ti2 input multiplexer . . . 2291
Table 518. TIMx internal trigger connection . . . 2292
Table 519. Timer break interconnect . . . 2292
Table 520. System break interconnect . . . 2293
Table 521. Interconnect to the ocref_clr input multiplexer . . . 2294
Table 522. CCR and ARR register change dithering pattern . . . 2324
Table 523. CCR register change dithering pattern in center-aligned PWM mode . . . 2325
Table 524. Asymmetric PWM modes . . . 2326
Table 525. Break protection disarming conditions . . . 2338
Table 526. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 2346
Table 527. Counting direction versus encoder signals and polarity settings . . . 2350
Table 528. DMA request . . . 2363
Table 529. Effect of low-power modes on TIM15/TIM16/TIM17 . . . 2363
Table 530. Interrupt requests . . . 2364
Table 531. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) . . . 2384
Table 532. TIM15 register map and reset values . . . 2402
Table 533. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) . . . 2416
Table 534. TIM16/TIM17 register map and reset values . . . 2433
Table 535. STM32G56x/55x/54x/53x LPTIM features . . . 2437
Table 536. LPTIM1 input/output pins . . . 2438
Table 537. LPTIM1 internal signals . . . 2439
Table 538. LPTIM1 external trigger connections . . . 2439
Table 539. LPTIM1 input 1 connections . . . 2440
Table 540. LPTIM1 input 2 connections . . . 2440
Table 541. LPTIM1 input capture 1 connections . . . 2440
Table 542. LPTIM1 input capture 2 connections . . . 2440
Table 543. Prescaler division ratios . . . 2442
Table 544. Encoder counting scenarios . . . 2449
Table 545. Input capture Glitch filter latency (in counter step unit) . . . 2453
Table 546. Effect of low-power modes on the LPTIM . . . 2457
Table 547. Interrupt events . . . 2458
Table 548. LPTIM register map and reset values . . . 2479
Table 549. IWDG features . . . 2481
Table 550. IWDG delays versus actions . . . 2483
Table 551. IWDG internal input/output signals . . . 2483
Table 552. Effect of low power modes on IWDG . . . 2488
Table 553. IWDG interrupt request . . . 2490
Table 554. IWDG register map and reset values . . . 2496
Table 555. WWDG features . . . 2497
Table 556. WWDG internal input/output signals . . . 2498
Table 557. WWDG interrupt requests . . . 2501
Table 558. WWDG register map and reset values . . . 2503
Table 559. RTC input/output pins . . . 2506
Table 560. RTC internal input/output signals . . . 2506
Table 561. RTC interconnection . . . 2507
Table 562. RTC_OUT mapping . . . 2507

Table 563. Effect of low-power modes on RTC . . . 2523
Table 564. RTC pins functionality over modes . . . 2523
Table 565. Nonsecure interrupt requests . . . 2524
Table 566. Secure interrupt requests . . . 2524
Table 567. RTC register map and reset values . . . 2555
Table 568. TAMP input/output pins . . . 2560
Table 569. TAMP internal input/output signals . . . 2560
Table 570. TAMP interconnection . . . 2561
Table 571. RTC pin PC13 configuration . . . 2564
Table 572. Backup registers and BHK tamper protection . . . 2569
Table 573. Other device resource x tamper protection . . . 2569
Table 574. Active tamper output change period . . . 2572
Table 575. Minimum ATPER value . . . 2574
Table 576. Active tamper filtered pulse duration . . . 2575
Table 577. Effect of low-power modes on TAMP . . . 2576
Table 578. TAMP pins functionality over modes . . . 2576
Table 579. Interrupt requests . . . 2577
Table 580. TAMP register map and reset values . . . 2602
Table 581. I2C implementation . . . 2605
Table 582. I2C input/output pins . . . 2606
Table 583. I2C internal input/output signals . . . 2607
Table 584. Comparison of analog and digital filters . . . 2609
Table 585. I²C-bus and SMBus specification data setup and hold times . . . 2611
Table 586. I2C configuration . . . 2615
Table 587. I²C-bus and SMBus specification clock timings . . . 2626
Table 588. Timing settings for f_{I2CCLK} of 8 MHz . . . 2636
Table 589. Timing settings for f_{I2CCLK} of 16 MHz . . . 2636
Table 590. SMBus timeout specifications . . . 2638
Table 591. SMBus with PEC configuration . . . 2640
Table 592. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 2641
Table 593. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 2641
Table 594. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 2641
Table 595. Effect of low-power modes to I2C . . . 2651
Table 596. I2C interrupt requests . . . 2651
Table 597. I2C register map and reset values . . . 2667
Table 598. I3C peripheral controller/target features versus MIPI v1.1 . . . 2670
Table 599. I3C instantiation . . . 2673
Table 600. I3C wake-up . . . 2673
Table 601. I3C FIFOs implementation . . . 2673
Table 602. I3C interrupt(s) . . . 2673
Table 603. I3C input/output pins . . . 2674
Table 604. I3C internal input/output signals . . . 2674
Table 605. I3C register usage . . . 2687
Table 606. I3C registers/fields usage versus controller state . . . 2688
Table 607. I3C registers/fields usage versus target state . . . 2691
Table 608. List of supported I3C CCCs, as controller/target . . . 2694
Table 609. I3C controller error management . . . 2726
Table 610. I3C target error management . . . 2728
Table 611. Effect of low-power modes . . . 2732
Table 612. I3C register map and reset values . . . 2778
Table 613. Instance implementation on STM32G56x/55x/54x/53x . . . 2782
Table 614. USART/LPUART features . . . 2782

Table 615. USART/UART input/output pins . . . 2785
Table 616. USART internal input/output signals . . . 2786
Table 617. Noise detection from sampled data . . . 2798
Table 618. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 2802
Table 619. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 2802
Table 620. USART frame formats . . . 2807
Table 621. Effect of low-power modes on the USART . . . 2830
Table 622. USART interrupt requests . . . 2831
Table 623. USART register map and reset values . . . 2869
Table 624. Instance implementation on STM32G56x/55x/54x/53x . . . 2872
Table 625. USART/LPUART features . . . 2872
Table 626. LPUART input/output pins . . . 2874
Table 627. LPUART internal input/output signals . . . 2874
Table 628. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 2885
Table 629. Tolerance of the LPUART receiver . . . 2886
Table 631. Effect of low-power modes on the LPUART . . . 2898
Table 632. LPUART interrupt requests . . . 2899
Table 633. LPUART register map and reset values . . . 2925
Table 634. SPI features . . . 2928
Table 635. SPI/I2S input/output pins . . . 2931
Table 636. SPI internal input/output signals . . . 2931
Table 637. Effect of low-power modes on the SPI . . . 2958
Table 638. SPI wake-up and interrupt requests . . . 2959
Table 639. Bitfields usable in PCM/I2S mode . . . 2961
Table 640. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 . . . 2970
Table 641. Serial data line swapping . . . 2970
Table 642. CLKGEN programming examples for usual I2S frequencies . . . 2974
Table 643. I2S interrupt requests . . . 2983
Table 644. SPI register map and reset values . . . 3001
Table 645. CAN subsystem I/O signals . . . 3007
Table 646. CAN subsystem I/O pins . . . 3008
Table 647. DLC coding in FDCAN . . . 3011
Table 648. Possible configurations for frame transmission . . . 3025
Table 649. Rx FIFO element . . . 3028
Table 650. Rx FIFO element description . . . 3028
Table 651. Tx buffer and FIFO element . . . 3030
Table 652. Tx buffer element description . . . 3030
Table 653. Tx event FIFO element . . . 3032
Table 654. Tx event FIFO element description . . . 3032
Table 655. Standard message ID filter element . . . 3033
Table 656. Standard message ID filter element field description . . . 3034
Table 657. Extended message ID filter element . . . 3034
Table 658. Extended message ID filter element field description . . . 3035
Table 659. FDCAN register map and reset values . . . 3065
Table 660. POSENC input/output pins . . . 3070
Table 661. POSENC internal input/output signals . . . 3071
Table 662. POSENC strobe sources . . . 3071
Table 663. POSENC clock requirements . . . 3072
Table 664. Hiperface-DSL Address conversion table . . . 3074
Table 665. Registers accesses in event-driven mode . . . 3081
Table 666. Fast position and velocity registers freeze in event-driven mode . . . 3083
Table 667. Safe position and CRC registers freeze in event-driven mode . . . 3083

Table 668. Strobe interface pulse-width requirements . . . 3085
Table 669. Interrupt summary . . . 3089
Table 670. POSENC register map and reset values . . . 3106
Table 671. STM32G55x/56x USB implementation . . . 3108
Table 672. USB input/output pins . . . 3110
Table 673. Double-buffering buffer flag definition . . . 3122
Table 674. Bulk double-buffering memory buffers usage (Device mode) . . . 3122
Table 675. Bulk double-buffering memory buffers usage (Host mode) . . . 3124
Table 676. Isochronous memory buffers usage . . . 3125
Table 677. Isochronous memory buffers usage . . . 3126
Table 678. Resume event detection . . . 3128
Table 679. Resume event detection for host . . . 3129
Table 680. Reception status encoding . . . 3147
Table 681. Endpoint/channel type encoding . . . 3147
Table 682. Endpoint/channel kind meaning . . . 3147
Table 683. Transmission status encoding . . . 3147
Table 684. USB register map and reset values . . . 3148
Table 685. Definition of allocated buffer memory . . . 3151
Table 686. USBSRAM register map and reset values . . . 3154
Table 687. UCPD implementation . . . 3156
Table 688. UCPD software trim data . . . 3156
Table 689. UCPD signals on pins . . . 3157
Table 690. UCPD internal signals . . . 3158
Table 691. 4b5b symbol encoding table . . . 3160
Table 692. Ordered sets . . . 3161
Table 693. Validation of ordered sets . . . 3161
Table 694. Data size . . . 3162
Table 695. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx . . . 3170
Table 696. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2) . . . 3172
Table 697. Effect of low power modes on the UCPD . . . 3174
Table 698. UCPD interrupt requests . . . 3175
Table 699. UCPD register map and reset values . . . 3191
Table 700. JTAG/Serial-wire debug port pins . . . 3195
Table 701. Trace port pins . . . 3195
Table 702. Single-wire trace port pins . . . 3196
Table 703. Authentication signal states with TrustZone® enabled (TZEN = 0xB4) . . . 3197
Table 704. Authentication signal states with TrustZone® disabled (TZEN = 0xC3) . . . 3198
Table 705. Life cycle state and debug states . . . 3198
Table 706. Definition of data to provision . . . 3201
Table 707. Permission mask (Endianness: Little Endian) . . . 3201
Table 708. JTAG-DP data registers . . . 3204
Table 709. Packet request . . . 3206
Table 710. ACK response . . . 3206
Table 711. Data transfer . . . 3206
Table 712. Debug port registers . . . 3207
Table 713. Debug port register map and reset values . . . 3213
Table 714. MEM-AP registers . . . 3215
Table 715. Access port register map and reset values . . . 3221
Table 716. System ROM table . . . 3222
Table 717. MCU ROM table . . . 3223
Table 718. Processor ROM table . . . 3223
Table 719. System ROM table register map and reset values . . . 3229

Table 720. MCU ROM table register map and reset values . . . 3234
Table 721. CPU ROM table register map and reset values . . . 3239
Table 722. DWT register map and reset values . . . 3255
Table 723. ITM register map and reset values . . . 3266
Table 724. BPU register map and reset values . . . 3274
Table 725. ETM register map and reset values . . . 3299
Table 726. TPIU register map and reset values . . . 3313
Table 727. CTI inputs . . . 3315
Table 728. CTI outputs . . . 3315
Table 729. CTI register map and reset values . . . 3326
Table 730. Peripheral clock freeze control bits . . . 3329
Table 731. Peripheral behavior in debug mode . . . 3330
Table 732. DBGMCU register map and reset values . . . 3345
Table 733. Document revision history . . . 3355

List of figures

Figure 1. System architecture . . . 115
Figure 2. Memory map based on IDAU mapping (STM32G55x/56x devices) . . . 123
Figure 3. Memory map based on IDAU mapping (STM32G53x/54x devices) . . . 124
Figure 4. Secure/nonsecure partitioning using TrustZone technology . . . 135
Figure 5. Sharing memory map between CPU in secure and nonsecure state . . . 137
Figure 6. Secure world transition and memory partitioning . . . 138
Figure 7. Global TrustZone framework and TrustZone awareness . . . 139
Figure 8. Flash memory TrustZone protections . . . 142
Figure 9. Device life-cycle security . . . 158
Figure 10. PRODUCT_STATEs (full TrustZone activated view) . . . 161
Figure 11. External Flash memory protection using SFI . . . 163
Figure 12. GTZC in Armv8-M subsystem block diagram . . . 170
Figure 13. GTZC block diagram . . . 172
Figure 14. Watermark memory protection controller (region x/subregions A and B) . . . 174
Figure 15. MPCBB block diagram . . . 175
Figure 16. FLASH block diagram (simplified) . . . 243
Figure 17. Embedded flash memory organization (512kbyte) . . . 245
Figure 18. Embedded flash memory usage . . . 246
Figure 19. Flash high-cycle data memory map on STM32G55xB/56xB devices (128 Kbytes) . . . 262
Figure 20. Flash high-cycle data memory map on STM32G55xC/56xC devices (256 Kbytes) . . . 262
Figure 21. Flash high-cycle data memory map on STM32G55xE/56xE devices (512 Kbytes) . . . 263
Figure 22. Flash high-cycle data memory map on STM32G53x8/54x8 devices (64 Kbytes) . . . 263
Figure 23. Flash high-cycle data memory map on STM32G53xB/54xB devices (128 Kbytes) . . . 264
Figure 24. Flash high-cycle data memory map on STM32G53xC/54xC devices (256 Kbytes) . . . 264
Figure 25. Flash bank swapping sequence . . . 269
Figure 26. OBK protection checks . . . 282
Figure 27. Key writing flow . . . 283
Figure 28. Swap workflow . . . 285
Figure 29. HDP example in user flash memory . . . 290
Figure 30. Protection attributes in case of bank swap illustration . . . 295
Figure 31. ICACHE block diagram (master2 port only on STM32G55x/56x . . . 380
Figure 32. ICACHE TAG and data memories functional view . . . 382
Figure 33. ICACHE remapping address mechanism . . . 386
Figure 34. DCACHE TAG and data memories functional view . . . 402
Figure 35. Power supply with SMPS . . . 422
Figure 36. Power supply with LDO . . . 423
Figure 37. System supply configurations . . . 425
Figure 38. Startup sequence if VCORE provided from external supply (LDO bypass) . . . 426
Figure 39. Power-on (POR) / power-down (PDR) reset waveform . . . 430
Figure 40. PVD thresholds . . . 431
Figure 41. AVD thresholds . . . 432
Figure 42. VDD monitoring threshold . . . 433

Figure 43. Temperature thresholds . . . 434
Figure 44. Dynamic voltage scaling in Run mode . . . 435
Figure 45. I/O states in Standby mode . . . 445
Figure 46. RCC block diagram . . . 467
Figure 47. Simplified reset circuit . . . 470
Figure 48. PLAY1 reset . . . 473
Figure 49. Top-level clock tree . . . 477
Figure 50. HSE configurations . . . 479
Figure 51. HSE clock generation . . . 479
Figure 52. HSI oscillator . . . 481
Figure 53. HSI48 oscillator . . . 483
Figure 54. LSE configurations . . . 484
Figure 55. LSE clock generation . . . 484
Figure 56. LSE and HSE CSS function . . . 487
Figure 57. PLL block diagram . . . 491
Figure 58. System clock generation . . . 497
Figure 59. Typical peripheral clock distribution . . . 500
Figure 60. Clock distribution for UARTs, USARTs and LPUART . . . 509
Figure 61. Clock distribution for SPIs . . . 510
Figure 62. Clock distribution for I2Cs . . . 511
Figure 63. Clock distribution for TIMs, HRTIM1, and LPTIM1 . . . 514
Figure 64. Clock distribution for POSENC1 sand POSENC2 . . . 515
Figure 65. Clock distribution for PLAY1 . . . 516
Figure 66. Clock distribution for ADCs and DACs . . . 519
Figure 67. Clock distribution for CCB subsystem . . . 521
Figure 68. Clock distribution for PKA, RNG, HASH and AES . . . 522
Figure 69. Clock distribution for FDCAN subsystem . . . 523
Figure 70. Clock distribution for XSPI1 (1) (2) . . . 524
Figure 71. Clock distribution for USB and UCPD1 . . . 525
Figure 72. Clock distribution for RTC/TAMP . . . 526
Figure 73. Clock distribution for WWDG and IWDG . . . 528
Figure 74. I/O compensation cell . . . 529
Figure 75. Debug and trace clock distribution . . . 530
Figure 76. Clock enable logic for peripherals . . . 533
Figure 77. Interrupt block diagram . . . 542
Figure 78. CRS block diagram . . . 653
Figure 79. CRS counter behavior . . . 655
Figure 80. Structure of 3- or 5-V tolerant GPIO (TT or FT) . . . 665
Figure 81. Input floating/pull-up/pull-down configurations . . . 669
Figure 82. Output configuration . . . 670
Figure 83. Alternate function configuration . . . 671
Figure 84. High-impedance analog configuration . . . 671
Figure 85. SBS block diagram . . . 686
Figure 86. Compensation cell management . . . 688
Figure 87. Compensation cell use . . . 688
Figure 88. SBS boot control . . . 689
Figure 89. SBS debug control . . . 693
Figure 90. SBS hardware secure storage control . . . 695
Figure 91. GPDMA block diagram . . . 739
Figure 92. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) . . . 740
Figure 93. GPDMA channel suspend and resume sequence . . . 741
Figure 94. GPDMA channel abort and restart sequence . . . 742

Figure 95. GPDMA channel terminate and drain FIFO sequence . . . 743
Figure 96. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 745
Figure 97. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 746
Figure 98. GPDMA dynamic linked-list data structure of a linear addressing channel x . . . 747
Figure 99. GPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 747
Figure 100. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) . . . 749
Figure 101. Inserting a LLIn with an auxiliary GPDMA channel y . . . 751
Figure 102. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) . . . 753
Figure 103. Building LLIn+1: GPDMA dynamic linked-lists in link step mode . . . 754
Figure 104. Replace with a new LLIn' in register file in link step mode . . . 755
Figure 105. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 756
Figure 106. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 757
Figure 107. GPDMA channel execution and linked-list programming . . . 759
Figure 108. Programmed 2D addressing . . . 762
Figure 109. GPDMA arbitration policy . . . 769
Figure 110. Trigger hit, memorization, and overrun waveform . . . 772
Figure 111. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 773
Figure 112. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 774
Figure 113. GPDMA block diagram . . . 822
Figure 114. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) . . . 823
Figure 115. GPDMA channel suspend and resume sequence . . . 824
Figure 116. GPDMA channel abort and restart sequence . . . 825
Figure 117. GPDMA channel terminate and drain FIFO sequence . . . 826
Figure 118. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 828
Figure 119. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 829
Figure 120. GPDMA dynamic linked-list data structure of a linear addressing channel x . . . 830
Figure 121. GPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 830
Figure 122. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) . . . 832
Figure 123. Inserting a LLIn with an auxiliary GPDMA channel y . . . 834
Figure 124. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) . . . 836
Figure 125. Building LLIn+1: GPDMA dynamic linked-lists in link step mode . . . 837
Figure 126. Replace with a new LLIn' in register file in link step mode . . . 838
Figure 127. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 839
Figure 128. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 840
Figure 129. GPDMA channel execution and linked-list programming . . . 842
Figure 130. Programmed 2D addressing . . . 845
Figure 131. GPDMA arbitration policy . . . 852
Figure 132. Trigger hit, memorization, and overrun waveform . . . 855

Figure 133. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 856
Figure 134. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 857
Figure 135. EXTI block diagram . . . 904
Figure 136. Configurable event trigger logic CPU wake-up . . . 907
Figure 137. EXTI direct events . . . 908
Figure 138. EXTI mux GPIO selection . . . 909
Figure 139. CRC calculation unit block diagram . . . 939
Figure 140. CORDIC convergence for trigonometric functions . . . 953
Figure 141. CORDIC convergence for hyperbolic functions . . . 954
Figure 142. CORDIC convergence for square root . . . 955
Figure 143. Block diagram . . . 964
Figure 144. Input buffer areas . . . 966
Figure 145. Circular input buffer . . . 967
Figure 146. Circular input buffer operation . . . 968
Figure 147. Circular output buffer . . . 969
Figure 148. Circular output buffer operation . . . 970
Figure 149. FIR filter structure . . . 972
Figure 150. IIR filter structure (direct form 1) . . . 974
Figure 151. X1 buffer initialization . . . 979
Figure 152. Filtering example 1 . . . 980
Figure 153. Filtering example 2 . . . 981
Figure 154. PLAY block diagram . . . 992
Figure 155. Input multiplexing . . . 999
Figure 156. Input filter and edge detection . . . 1000
Figure 157. Filter operation example . . . 1000
Figure 158. Edge detector operation (rising and falling edge) with filter enabled . . . 1001
Figure 159. Edge detector operation (rising and falling edge) with filter bypassed . . . 1001
Figure 160. swin_we pulse timing . . . 1002
Figure 161. Logic element . . . 1003
Figure 162. Example logic expression 1 . . . 1003
Figure 163. Example logic expression 2 . . . 1004
Figure 164. Input interconnect . . . 1006
Figure 165. Logic element cascading . . . 1009
Figure 166. Combining logic elements in parallel . . . 1009
Figure 167. Splitting logic expressions between logic elements . . . 1010
Figure 168. Output interconnect . . . 1011
Figure 169. XSPI block diagram for octal configuration . . . 1033
Figure 170. XSPI block diagram for quad configuration . . . 1033
Figure 171. XSPI block diagram for dual-quad configuration . . . 1034
Figure 172. SDR read command in octal configuration . . . 1035
Figure 173. DTR read in octal-SPI mode with DQS (Macronix mode) example . . . 1038
Figure 174. SDR write command in octal-SPI mode example . . . 1040
Figure 175. DTR write in octal-SPI mode (Macronix mode) example . . . 1041
Figure 176. Example of HyperBus read operation (8-bit data mode) . . . 1042
Figure 177. HyperBus write operation with initial latency . . . 1043
Figure 178. HyperBus read operation with additional latency . . . 1044
Figure 179. HyperBus write operation with additional latency . . . 1044
Figure 180. HyperBus write operation with no latency (register write) . . . 1045
Figure 181. HyperBus read operation page crossing with latency . . . 1045
Figure 182. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access . . . 1052

Figure 183. D0/D1 data ordering in octal-SPI DTR mode with DQS and CLK polarity aligned during Read access . . . 1052
Figure 184. OctaRAM read operation with reverse data ordering D1/D0 . . . 1053
Figure 185. NCS when CKMODE = 0 (T = CLK period) . . . 1058
Figure 186. Example of software control of two external memories (octal mode) . . . 1059
Figure 187. Example of hardware-controller extended memory support (octal mode) . . . 1060
Figure 188. DLYB block diagram . . . 1089
Figure 189. ADC block diagram . . . 1097
Figure 190. ADC clock scheme . . . 1103
Figure 191. ADC1 connectivity . . . 1105
Figure 192. ADC2 connectivity . . . 1106
Figure 193. ADC3 connectivity . . . 1107
Figure 194. ADC calibration . . . 1111
Figure 195. Updating the ADC calibration factor . . . 1111
Figure 196. Mixing single-ended and differential channels . . . 1112
Figure 197. Enabling/disabling the ADC . . . 1113
Figure 198. Bulb mode timing diagram . . . 1116
Figure 199. Analog-to-digital conversion time . . . 1121
Figure 200. Stopping ongoing regular conversions . . . 1122
Figure 201. Stopping ongoing regular and injected conversions . . . 1123
Figure 202. Triggers shared between ADC master and slave . . . 1124
Figure 203. Injected conversion latency during ongoing regular conversion . . . 1125
Figure 204. Example of ADC_JSQR queue of context (sequence change) . . . 1128
Figure 205. Example of ADC_JSQR queue of context (trigger change) . . . 1128
Figure 206. Example of ADC_JSQR queue of context with overflow before conversion . . . 1129
Figure 207. Example of ADC_JSQR queue of context with overflow during conversion . . . 1129
Figure 208. Example of ADC_JSQR queue of context with empty queue (case JQM = 0) . . . 1130
Figure 209. Example of ADC_JSQR queue of context with empty queue (JQM = 1) . . . 1131
Figure 210. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion . . . 1131
Figure 211. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs . . . 1132
Figure 212. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion . . . 1132
Figure 213. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 1) . . . 1133
Figure 214. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 0) . . . 1133
Figure 215. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 1) . . . 1134
Figure 216. Single conversions of a sequence, software trigger . . . 1136
Figure 217. Continuous conversion of a sequence, software trigger . . . 1136
Figure 218. Single conversions of a sequence, hardware trigger . . . 1137
Figure 219. Continuous conversions of a sequence, hardware trigger . . . 1137
Figure 220. Right alignment (offset disabled, unsigned value) . . . 1139
Figure 221. Right alignment (offset enabled, signed value) . . . 1140
Figure 222. Left alignment (offset disabled, unsigned value) . . . 1140
Figure 223. Left alignment (offset enabled, signed value) . . . 1141
Figure 224. Example of overrun (OVRMOD = 0) . . . 1144
Figure 225. Example of overrun (OVRMOD = 1) . . . 1144
Figure 226. AUTDLY = 1, regular conversion in continuous mode, software trigger . . . 1147
Figure 227. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 1148
Figure 228. AUTDLY = 1, regular hardware conversions interrupted by injected conversions . . .

(DISCEN = 1, JDISCEN = 1) . . . 1149
Figure 229. AUTDLY = 1, regular continuous conversions interrupted by injected conversions 1149
Figure 230. AUTDLY = 1 in autoinjection mode (JAUTO = 1) . . . 1150
Figure 231. Analog watchdog guarded area . . . 1151
Figure 232. ADC_AWDy_OUT signal generation (on all regular channels). . . . 1153
Figure 233. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software) . . . 1154
Figure 234. ADC_AWDy_OUT signal generation (on a single regular channel) . . . 1154
Figure 235. ADC_AWDy_OUT signal generation (on all injected channels) . . . 1155
Figure 236. 12-bit result oversampling with 10-bit right shift and rounding . . . 1156
Figure 237. Triggered regular oversampling mode (TROVS bit = 1) . . . 1158
Figure 238. Regular oversampling modes (4x ratio) . . . 1159
Figure 239. Regular and injected oversampling modes used simultaneously . . . 1160
Figure 240. Triggered regular oversampling with injection . . . 1160
Figure 241. Oversampling in autoinjection mode. . . . 1161
Figure 242. Triggered injected parallel oversampling for 3 channels example . . . 1162
Figure 243. Injected oversampling with intermediate value enabled example. . . . 1163
Figure 244. Triggered injected parallel oversampling for 3 channels with accumulated data example . . . 1163
Figure 245. Dual ADC block diagram . . . 1165
Figure 246. Injected simultaneous mode on four channels: Dual ADC mode . . . 1166
Figure 247. Regular simultaneous mode on 16 channels: dual ADC mode . . . 1168
Figure 248. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode . . . 1170
Figure 249. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode . . . 1170
Figure 250. Interleaved conversion with injection . . . 1171
Figure 251. Alternate trigger: injected group of each ADC . . . 1172
Figure 252. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode . . . 1173
Figure 253. Alternate + regular simultaneous . . . 1173
Figure 254. Case of trigger occurring during injected conversion . . . 1174
Figure 255. Interleaved single channel CH0 with injected sequence CH11, CH12 . . . 1174
Figure 256. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 case 1: Master interrupted first . . . 1175
Figure 257. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 case 2: Slave interrupted first . . . 1175
Figure 258. DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00 . . . 1176
Figure 259. DMA requests in interleaved mode when DAMDF = 0b10 . . . 1177
Figure 260. Temperature sensor channel block diagram . . . 1179
Figure 261. VREFINT channel block diagram . . . 1181
Figure 262. Temperature sensor functional block diagram . . . 1229
Figure 263. Method for low REF_CLK frequencies . . . 1231
Figure 264. Method for high REF_CLK frequencies . . . 1231
Figure 265. Temperature sensor sequence. . . . 1234
Figure 266. DAC block diagram . . . 1247
Figure 267. Data registers in single DAC channel mode . . . 1251
Figure 268. Timing diagram for conversion with trigger disabled TEN = 0 . . . 1253
Figure 269. DAC LFSR register calculation algorithm . . . 1255
Figure 270. DAC conversion (SW trigger enabled) with LFSR wave generation. . . . 1255
Figure 271. DAC triangle wave generation . . . 1256
Figure 272. DAC conversion (SW trigger enabled) with triangle wave generation . . . 1256
Figure 273. DAC sawtooth wave generation (STDIRx = 0) . . . 1257
Figure 274. DAC sawtooth wave generation (STDIRx = 1) . . . 1257

Figure 275. DAC sawtooth STINCTRIG and STRSTTRIG priority (STDIR = 0) 1258
Figure 276. DAC extended sawtooth wave generation (STDIR1 = 0) 1260
Figure 277. DAC extended sawtooth wave generation (STDIR1 = 1) 1261
Figure 278. DAC sample and hold mode phase diagram 1264
Figure 279. VREFBUF block diagram 1281
Figure 280. Comparator simplified functional block diagram 1287
Figure 281. Window comparators functional block diagram (x=1,3,5,7...) 1290
Figure 282. Comparator hysteresis 1290
Figure 283. Comparator output digital filter (N=4) 1291
Figure 284. Comparator output digital filter (N=4) 1291
Figure 285. Comparator output blanking 1292
Figure 286. OPAMP block diagram 1300
Figure 287. Stand-alone operational amplifier 1304
Figure 288. Follower 1305
Figure 289. Noninverting PGA 1306
Figure 290. Noninverting PGA with filtering 1307
Figure 291. Noninverting PGA with external bias, or inverting PGA 1308
Figure 292. Inverting or noninverting PGA with external bias or/and filtering 1309
Figure 293. Simplified view 1309
Figure 294. Timer-controlled operation for a dual motor application 1312
Figure 295. CCB block diagram 1322
Figure 296. CCB operation CPU → PKA + SAES (write) 1324
Figure 297. CCB operation PKA → SAES 1324
Figure 298. CCB operation SAES → PKA 1324
Figure 299. CCB operation SAES → CCB 1325
Figure 300. CCB operation RNG → PKA (write) 1325
Figure 301. CCB operation RNG → SAES (write) 1326
Figure 302. Blob creation scenarios 1327
Figure 303. RNG block diagram 1348
Figure 304. NIST SP800-90B entropy source model 1349
Figure 305. RNG initialization overview 1352
Figure 306. AES block diagram 1369
Figure 307. Encryption/ decryption typical usage 1370
Figure 308. Typical operation with authentication 1373
Figure 309. Example of suspend mode management 1374
Figure 310. ECB encryption 1375
Figure 311. ECB decryption 1375
Figure 312. CBC encryption 1375
Figure 313. CBC decryption 1376
Figure 314. Message construction in CTR mode 1378
Figure 315. CTR encryption 1379
Figure 316. Message construction in GCM 1380
Figure 317. GCM authenticated encryption 1382
Figure 318. Message construction in GMAC mode 1385
Figure 319. GMAC authentication mode 1386
Figure 320. Message construction in CCM mode 1387
Figure 321. CCM mode authenticated encryption 1388
Figure 322. 128-bit block construction according to the data type 1393
Figure 323. SAES block diagram 1415
Figure 324. Encryption/ decryption typical usage 1417
Figure 325. Typical operation with authentication 1419
Figure 326. Example of suspend mode management 1420

Figure 327. ECB encryption . . . 1421
Figure 328. ECB decryption . . . 1421
Figure 329. CBC encryption . . . 1422
Figure 330. CBC decryption . . . 1422
Figure 331. Message construction in CTR mode . . . 1425
Figure 332. CTR encryption . . . 1426
Figure 333. Message construction in GCM . . . 1427
Figure 334. GCM authenticated encryption . . . 1429
Figure 335. Message construction in GMAC mode . . . 1432
Figure 336. GMAC authentication mode . . . 1432
Figure 337. Message construction in CCM mode . . . 1433
Figure 338. CCM mode authenticated encryption . . . 1435
Figure 339. Operation with wrapped keys for SAES in ECB and CBC modes . . . 1438
Figure 340. Operation with wrapped keys for SAES in CTR mode . . . 1441
Figure 341. Usage of Shared-key mode . . . 1442
Figure 342. 128-bit block construction according to the data type . . . 1445
Figure 343. Key protection mechanisms . . . 1447
Figure 344. HASH block diagram . . . 1469
Figure 345. Message data swapping feature . . . 1471
Figure 346. HASH suspend/resume mechanism . . . 1477
Figure 347. PKA block diagram . . . 1491
Figure 348. PKA block diagram . . . 1522
Figure 349. Chaining of PKA with SAES or RNG . . . 1529
Figure 350. OTFDEC block diagram . . . 1561
Figure 351. Typical OTFDEC use in a SoC . . . 1562
Figure 352. AES CTR decryption flow . . . 1563
Figure 353. OTFDEC flow control overview (dual burst read request) . . . 1564
Figure 354. High-resolution timer overview . . . 1586
Figure 355. Counter and capture register format vs clock prescaling factor . . . 1595
Figure 356. Timer A..F overview . . . 1597
Figure 357. Continuous timer operation . . . 1599
Figure 358. Single-shot timer operation . . . 1599
Figure 359. Timer reset resynchronization (prescaling ratio above 32) . . . 1601
Figure 360. Repetition rate versus HRTIM_REPxR content in continuous mode . . . 1602
Figure 361. Repetition counter behavior in single-shot mode . . . 1602
Figure 362. Compare events action on outputs: set on compare 1, reset on compare 2 . . . 1604
Figure 363. Timer A timing unit capture circuitry . . . 1608
Figure 364. Auto-delayed overview (compare 2 only) . . . 1609
Figure 365. Auto-delayed compare . . . 1610
Figure 366. Triggered-half mode example . . . 1612
Figure 367. Triggered-half mode timeout . . . 1613
Figure 368. Push-pull mode block diagram . . . 1614
Figure 369. Push-pull mode example . . . 1615
Figure 370. Push-Pull with deadtime . . . 1616
Figure 371. Deadtime generation . . . 1617
Figure 372. Complementary outputs with deadtime insertion . . . 1617
Figure 373. Deadtime insertion versus deadtime sign (1 indicates negative deadtime) . . . 1618
Figure 374. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 0) . . . 1619
Figure 375. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 1) . . . 1619
Figure 376. Complementary outputs for low pulsewidth (SDTRx = 0, SDTFx = 1) . . . 1619
Figure 377. Complementary outputs for low pulsewidth (SDTRx = 1, SDTFx=0) . . . 1620
Figure 378. Deadtime insertion mode . . . 1620

Figure 379. Deadtime insertion with cycle-by-cycle overcurrent protection . . . 1621
Figure 380. Deadtime insertion example with output set on external event . . . 1621
Figure 381. Deadtime insertion: output priority example . . . 1622
Figure 382. Deadtime insertion and disarm . . . 1622
Figure 383. Deadtime insertion and disarm example 1 . . . 1623
Figure 384. Deadtime insertion and disarm example 2 . . . 1623
Figure 385. Crossbar override mode . . . 1624
Figure 386. Delayed fault operation . . . 1626
Figure 387. Deadtime on Fault . . . 1626
Figure 388. Crossbar override force on event mode . . . 1627
Figure 389. Delayed Fault exit example . . . 1628
Figure 390. Master timer overview . . . 1629
Figure 391. Basic symmetric waveform in up-down counting mode . . . 1630
Figure 392. Complex symmetric waveform in up-down counting mode . . . 1631
Figure 393. Asymmetric waveform in up-down counting mode . . . 1631
Figure 394. External event management in up-down counting mode . . . 1632
Figure 395. Interleaved up-down counter operation . . . 1632
Figure 396. Interleaved up-down counter operation . . . 1633
Figure 397. Push-pull up-down mode example . . . 1633
Figure 398. Up-down mode with “greater than” comparison . . . 1634
Figure 399. Up-down mode with output set on period event, for OUTROM[1:0]=10 . . . 1635
Figure 400. Repetition counter behavior in up-down counting mode . . . 1636
Figure 401. Short distance set/reset management for narrow pulse generation . . . 1638
Figure 402. External event conditioning overview (1 channel represented) . . . 1641
Figure 403. Latency to external events (counter reset and output set) . . . 1645
Figure 404. Latency to external events (output reset on external event) . . . 1645
Figure 405. Event blanking mode . . . 1646
Figure 406. Event postpone mode . . . 1646
Figure 407. External event blanking with edge-sensitive trigger . . . 1647
Figure 408. External event blanking, level sensitive triggering . . . 1648
Figure 409. Push-pull blanking mode . . . 1649
Figure 410. Event windowing mode . . . 1649
Figure 411. External event windowing with edge-sensitive trigger . . . 1651
Figure 412. External event windowing, level sensitive triggering . . . 1651
Figure 413. External event counters . . . 1652
Figure 414. External event counter cumulative mode (EEVxRSTM = 1, EEVxCNT = 2) . . . 1653
Figure 415. Delayed Idle mode entry . . . 1654
Figure 416. Burst mode and delayed protection priorities (DIDL = 0) . . . 1655
Figure 417. Burst mode and delayed protection priorities (DIDL = 1) . . . 1656
Figure 418. Balanced Idle protection example . . . 1657
Figure 419. Resynchronized timer update (TAU=1 in HRTIM_TIMBCR) . . . 1662
Figure 420. Early turn-ON and early turn-OFF behavior in “greater than” PWM mode . . . 1664
Figure 421. Output management overview . . . 1669
Figure 422. HRTIM output states and transitions . . . 1670
Figure 423. Burst mode operation example . . . 1671
Figure 424. Burst mode trigger on external event . . . 1674
Figure 425. Delayed burst mode entry with deadtime enabled and IDLESx = 1 . . . 1675
Figure 426. Delayed burst mode entry during deadtime . . . 1676
Figure 427. Burst mode exit when the deadtime generator is enabled . . . 1677
Figure 428. Burst mode emulation example . . . 1679
Figure 429. Carrier frequency signal insertion . . . 1679
Figure 430. HRTIM outputs with Chopper mode enabled . . . 1680

Figure 431. Fault protection circuitry (FAULT1 fully represented, FAULT2..8 partially). 1681
Figure 432. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING}$ = fHRTIM, N = 4) 1682
Figure 433. Fault counter cumulative mode (FLTxRSTM = 1, FLTxCNT[3:0] = 2) 1684
Figure 434. Auxiliary outputs 1686
Figure 435. Auxiliary and main outputs during burst mode (DIDLx = 0) 1687
Figure 436. Deadtime distortion on auxiliary output when exiting burst mode. 1687
Figure 437. Counter behavior in synchronized start mode 1691
Figure 438. ADC trigger selection overview 1692
Figure 439. ADC triggers 1693
Figure 440. ADC trigger post-scaling in up-counting mode 1694
Figure 441. ADC trigger post-scaling in up/down counting mode 1694
Figure 442. Combining several updates on a single hrtim_dac_trgx output 1695
Figure 443. DAC triggers for slope compensation 1698
Figure 444. DAC triggers overview 1699
Figure 445. DMA burst overview 1703
Figure 446. Burst DMA operation flowchart 1704
Figure 447. Registers update following DMA burst transfer 1705
Figure 448. Buck converter topology 1708
Figure 449. Dual Buck converter management 1708
Figure 450. Synchronous rectification depending on output current 1709
Figure 451. Buck with synchronous rectification 1709
Figure 452. 3-phase interleaved buck converter 1710
Figure 453. 3-phase interleaved buck converter control 1711
Figure 454. Transition mode PFC 1711
Figure 455. Transition mode PFC waveforms 1712
Figure 456. Advanced-control timer block diagram 1890
Figure 457. Counter timing diagram with prescaler division change from 1 to 2 1899
Figure 458. Counter timing diagram with prescaler division change from 1 to 4 1899
Figure 459. Counter timing diagram, internal clock divided by 1 1901
Figure 460. Counter timing diagram, internal clock divided by 2 1901
Figure 461. Counter timing diagram, internal clock divided by 4 1902
Figure 462. Counter timing diagram, internal clock divided by N. 1902
Figure 463. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded). 1903
Figure 464. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded). 1904
Figure 465. Counter timing diagram, internal clock divided by 1 1905
Figure 466. Counter timing diagram, internal clock divided by 2 1906
Figure 467. Counter timing diagram, internal clock divided by 4 1906
Figure 468. Counter timing diagram, internal clock divided by N. 1907
Figure 469. Counter timing diagram, update event when repetition counter is not used. 1907
Figure 470. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 1909
Figure 471. Counter timing diagram, internal clock divided by 2 1909
Figure 472. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 1910
Figure 473. Counter timing diagram, internal clock divided by N. 1910
Figure 474. Counter timing diagram, update event with ARPE = 1 (counter underflow) 1911
Figure 475. Counter timing diagram, Update event with ARPE = 1 (counter overflow). 1912
Figure 476. Update rate examples depending on mode and TIM_RCR register settings 1913
Figure 477. External trigger input block 1914
Figure 478. Control circuit in normal mode, internal clock divided by 1 1915
Figure 479. tim_ti2 external clock connection example 1915
Figure 480. Control circuit in external clock mode 1 1916

Figure 481. External trigger input block . . . 1917
Figure 482. Control circuit in external clock mode 2 . . . 1918
Figure 483. Capture/compare channel (example: channel 1 input stage) . . . 1918
Figure 484. Capture/compare channel 1 main circuit . . . 1919
Figure 485. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) . . . 1920
Figure 486. Output stage of capture/compare channel (channel 5, idem ch. 6 and 7) . . . 1920
Figure 487. PWM input mode timing . . . 1922
Figure 488. Output compare mode, toggle on tim_oc1 . . . 1924
Figure 489. Edge-aligned PWM waveforms (ARR = 8) . . . 1926
Figure 490. Center-aligned PWM waveforms (ARR = 8) . . . 1927
Figure 491. Dithering principle . . . 1928
Figure 492. Data format and register coding in dithering mode . . . 1929
Figure 493. PWM resolution vs frequency . . . 1930
Figure 494. PWM dithering pattern . . . 1931
Figure 495. Dithering effect on duty cycle in center-aligned PWM mode . . . 1932
Figure 496. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 1933
Figure 497. Asymmetric mode example for single-shunt current sensing . . . 1936
Figure 498. Asymmetric mode 8 example . . . 1936
Figure 499. Combined PWM mode examples . . . 1938
Figure 500. Combined PWM 3 example . . . 1939
Figure 501. 3-phase combined PWM signals with multiple trigger pulses per period . . . 1940
Figure 502. Complementary output with symmetrical dead-time insertion . . . 1941
Figure 503. Asymmetrical deadtime . . . 1942
Figure 504. Dead-time waveforms with delay greater than the negative pulse . . . 1942
Figure 505. Dead-time waveforms with delay greater than the positive pulse . . . 1942
Figure 506. Clock prescalers for dead-time and digital filters . . . 1943
Figure 507. Break and Break2 circuitry overview . . . 1945
Figure 508. Various output behavior in response to a break event on tim_brk (OSSI = 1) . . . 1947
Figure 509. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1) . . . 1948
Figure 510. PWM output state following tim_brk assertion (OSSI = 0) . . . 1949
Figure 511. Immediate break and Delayed 1/2 break modes . . . 1949
Figure 512. Pulse termination during delayed break . . . 1950
Figure 513. Delayed break with deadtime insertion . . . 1951
Figure 514. Output redirection (tim_brk2 request not represented) . . . 1953
Figure 515. tim_ocref_clr input selection multiplexer . . . 1954
Figure 516. Clearing TIMx tim_ocxref . . . 1955
Figure 517. Asymmetric PWM mode 1 . . . 1955
Figure 518. Asymmetric PWM mode 2 . . . 1956
Figure 519. Asymmetric PWM mode 7 (OC1refc = oc1ref OR oc5ref) . . . 1956
Figure 520. Combined PWM mode 1 . . . 1957
Figure 521. 6-step generation, COM example (OSSR = 1) . . . 1958
Figure 522. Example of one pulse mode. . . . 1959
Figure 523. Retriggerable one-pulse mode . . . 1960
Figure 524. Pulse generator circuitry . . . 1961
Figure 525. Pulse generation on compare event, for edge-aligned and encoder modes . . . 1962
Figure 526. Extended pulsewidth in case of concurrent triggers . . . 1963
Figure 527. Example of counter operation in encoder interface mode. . . . 1965
Figure 528. Example of encoder interface mode with tim_ti1fp1 polarity inverted. . . . 1965
Figure 529. Quadrature encoder counting modes . . . 1966
Figure 530. Direction plus clock encoder mode . . . 1967
Figure 531. Directional clock encoder mode (CC1P = CC2P = 0) . . . 1967
Figure 532. Directional clock encoder mode (CC1P = CC2P = 1) . . . 1968

Figure 533. Index gating options . . . 1969
Figure 534. Jittered Index signals . . . 1969
Figure 535. Index generation for IPOS[1:0] = 11 . . . 1970
Figure 536. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 1970
Figure 537. Counter reading with index ungated (IPOS[1:0] = 00) . . . 1971
Figure 538. Counter reading with index gated on channel A and B . . . 1971
Figure 539. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 1972
Figure 540. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 1973
Figure 541. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 1974
Figure 542. Directional index sensitivity . . . 1974
Figure 543. Counter reset as function of FIDX bit setting . . . 1975
Figure 544. Index blanking . . . 1975
Figure 545. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 1976
Figure 546. Index behavior in directional clock mode, IPOS[0] = 1 . . . 1976
Figure 547. State diagram for quadrature encoded signals . . . 1977
Figure 548. Up-counting encoder error detection . . . 1978
Figure 549. Down-counting encode error detection . . . 1979
Figure 550. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 1980
Figure 551. Encoder with built-in debouncer, x2 mode . . . 1981
Figure 552. Encoder with built-in debouncer, x4 mode . . . 1981
Figure 553. Clock drift measure using directional clock mode . . . 1982
Figure 554. Measuring time interval between edges on three signals . . . 1983
Figure 555. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 1984
Figure 556. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 1984
Figure 557. Example of Hall sensor interface . . . 1986
Figure 558. Control circuit in reset mode . . . 1987
Figure 559. Control circuit in Gated mode . . . 1988
Figure 560. Control circuit in trigger mode . . . 1989
Figure 561. Control circuit in external clock mode 2 + trigger mode . . . 1990
Figure 562. tim_trgo2 postscaler example . . . 1991
Figure 563. General-purpose timer block diagram . . . 2071
Figure 564. Counter timing diagram with prescaler division change from 1 to 2 . . . 2078
Figure 565. Counter timing diagram with prescaler division change from 1 to 4 . . . 2079
Figure 566. Counter timing diagram, internal clock divided by 1 . . . 2080
Figure 567. Counter timing diagram, internal clock divided by 2 . . . 2080
Figure 568. Counter timing diagram, internal clock divided by 4 . . . 2081
Figure 569. Counter timing diagram, internal clock divided by N . . . 2081
Figure 570. Counter timing diagram, Update event when ARPE = 0 (TIM_ARR not preloaded) . . . 2082
Figure 571. Counter timing diagram, Update event when ARPE = 1 (TIM_ARR preloaded) . . . 2083
Figure 572. Counter timing diagram, internal clock divided by 1 . . . 2084
Figure 573. Counter timing diagram, internal clock divided by 2 . . . 2085
Figure 574. Counter timing diagram, internal clock divided by 4 . . . 2085
Figure 575. Counter timing diagram, internal clock divided by N . . . 2086
Figure 576. Counter timing diagram, Update event . . . 2086
Figure 577. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 . . . 2088
Figure 578. Counter timing diagram, internal clock divided by 2 . . . 2088
Figure 579. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 . . . 2089
Figure 580. Counter timing diagram, internal clock divided by N . . . 2089
Figure 581. Counter timing diagram, Update event with ARPE = 1 (counter underflow) . . . 2090
Figure 582. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 2091
Figure 583. External trigger input block . . . 2091
Figure 584. Control circuit in normal mode, internal clock divided by 1 . . . 2093

Figure 585. tim_ti2 external clock connection example 2093
Figure 586. Control circuit in external clock mode 1 2094
Figure 587. External trigger input block 2095
Figure 588. Control circuit in external clock mode 2 2096
Figure 589. Capture/compare channel (example: channel 1 input stage) 2096
Figure 590. Capture/compare channel 1 main circuit 2097
Figure 591. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4) 2097
Figure 592. PWM input mode timing 2100
Figure 593. Output compare mode, toggle on tim_oc1 2102
Figure 594. Edge-aligned PWM waveforms (ARR = 8) 2103
Figure 595. Center-aligned PWM waveforms (ARR = 8) 2104
Figure 596. Dithering principle 2105
Figure 597. Data format and register coding in dithering mode 2106
Figure 598. PWM resolution vs frequency (16-bit mode) 2107
Figure 599. PWM resolution vs frequency (32-bit mode) 2107
Figure 600. PWM dithering pattern 2108
Figure 601. Dithering effect on duty cycle in center-aligned PWM mode 2109
Figure 602. Generation of two phase-shifted PWM signals with 50% duty cycle 2111
Figure 603. Combined PWM mode on channels 1 and 3 2113
Figure 604. Clock prescalers for dead-time and digital filters 2114
Figure 605. OCREF_CLR input selection multiplexer 2114
Figure 606. Clearing TIMx tim_ocxref 2115
Figure 607. Asymmetric PWM mode 1 2116
Figure 608. Asymmetric PWM mode 2 2116
Figure 609. Combined PWM mode 1 2117
Figure 610. Example of one-pulse mode 2118
Figure 611. Retriggerable one-pulse mode 2119
Figure 612. Pulse generator circuitry 2120
Figure 613. Pulse generation on compare event, for edge-aligned and encoder modes 2121
Figure 614. Extended pulse width in case of concurrent triggers 2122
Figure 615. Example of counter operation in encoder interface mode 2124
Figure 616. Example of encoder interface mode with tim_ti1fp1 polarity inverted 2124
Figure 617. Quadrature encoder counting modes 2125
Figure 618. Direction plus clock encoder mode 2126
Figure 619. Directional clock encoder mode (CC1P = CC2P = 0) 2127
Figure 620. Directional clock encoder mode (CC1P = CC2P = 1) 2127
Figure 621. Index gating options 2129
Figure 622. Jittered Index signals 2129
Figure 623. Index generation for IPOS[1:0] = 11 2130
Figure 624. Counter reading with index gated on channel A (IPOS[1:0] = 11) 2130
Figure 625. Counter reading with index ungated (IPOS[1:0] = 00) 2131
Figure 626. Counter reading with index gated on channel A and B 2131
Figure 627. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) 2132
Figure 628. Counter reset Narrow index pulse (closer view, ARR = 0x07) 2133
Figure 629. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) 2134
Figure 630. Directional index sensitivity 2134
Figure 631. Counter reset as function of FIDX bit setting 2135
Figure 632. Index blanking 2135
Figure 633. Index behavior in clock + direction mode, IPOS[0] = 1 2136
Figure 634. Index behavior in directional clock mode, IPOS[0] = 1 2136
Figure 635. State diagram for quadrature encoded signals 2137
Figure 636. Up-counting encoder error detection 2138

Figure 637. Down-counting encode error detection . . . 2139
Figure 638. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 2140
Figure 639. Encoder with built-in debouncer, x2 mode . . . 2141
Figure 640. Encoder with built-in debouncer, x4 mode . . . 2141
Figure 641. Clock drift measure using directional clock mode . . . 2142
Figure 642. Measuring time interval between edges on three signals . . . 2143
Figure 643. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 2144
Figure 644. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 2144
Figure 645. Control circuit in reset mode . . . 2145
Figure 646. Control circuit in gated mode . . . 2146
Figure 647. Control circuit in trigger mode . . . 2147
Figure 648. Control circuit in external clock mode 2 + trigger mode . . . 2148
Figure 649. Master/Slave timer example . . . 2149
Figure 650. Master/slave connection example with 1 channel only timers . . . 2149
Figure 651. Gating TIM_slv with tim_oc1ref of TIM_mstr . . . 2150
Figure 652. Gating TIM_slv with Enable of TIM_mstr . . . 2151
Figure 653. Triggering TIM_slv with update of TIM_mstr . . . 2152
Figure 654. Triggering TIM_slv with Enable of TIM_mstr . . . 2152
Figure 655. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input . . . 2153
Figure 656. Basic timer block diagram . . . 2198
Figure 657. Control circuit in normal mode, internal clock divided by 1 . . . 2199
Figure 658. Counter timing diagram with prescaler division change from 1 to 2 . . . 2200
Figure 659. Counter timing diagram with prescaler division change from 1 to 4 . . . 2201
Figure 660. Counter timing diagram, internal clock divided by 1 . . . 2202
Figure 661. Counter timing diagram, internal clock divided by 2 . . . 2202
Figure 662. Counter timing diagram, internal clock divided by 4 . . . 2203
Figure 663. Counter timing diagram, internal clock divided by N . . . 2203
Figure 664. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 2204
Figure 665. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 2205
Figure 666. Dithering principle . . . 2206
Figure 667. Data format and register coding in dithering mode . . . 2206
Figure 668. FCnt resolution vs frequency . . . 2207
Figure 669. PWM dithering pattern . . . 2207
Figure 670. General-purpose timer block diagram (TIM9/TIM12) . . . 2221
Figure 671. General-purpose timer block diagram (TIM10/TIM11/TIM13/TIM14) . . . 2222
Figure 672. Counter timing diagram with prescaler division change from 1 to 2 . . . 2226
Figure 673. Counter timing diagram with prescaler division change from 1 to 4 . . . 2226
Figure 674. Counter timing diagram, internal clock divided by 1 . . . 2227
Figure 675. Counter timing diagram, internal clock divided by 2 . . . 2228
Figure 676. Counter timing diagram, internal clock divided by 4 . . . 2228
Figure 677. Counter timing diagram, internal clock divided by N . . . 2229
Figure 678. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 2229
Figure 679. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 2230
Figure 680. Control circuit in normal mode, internal clock divided by 1 . . . 2231
Figure 681. tim_ti2 external clock connection example . . . 2231
Figure 682. Control circuit in external clock mode 1 . . . 2232
Figure 683. Capture/compare channel 1 input stage (TIM10/TIM11/TIM13/TIM14) . . . 2233
Figure 684. Capture/compare channel 1 input stage (TIM9/TIM12) . . . 2233

Figure 685. Capture/compare channel 1 main circuit . . . 2234
Figure 686. Output stage of capture/compare channel 1 . . . 2234
Figure 687. PWM input mode timing . . . 2236
Figure 688. Output compare mode, toggle on tim_oc1 . . . 2238
Figure 689. Edge-aligned PWM waveforms (ARR = 8) . . . 2239
Figure 690. Dithering principle . . . 2240
Figure 691. Data format and register coding in dithering mode . . . 2240
Figure 692. PWM resolution vs frequency . . . 2241
Figure 693. PWM dithering pattern . . . 2242
Figure 694. Combined PWM mode on channel 1 and 2 . . . 2244
Figure 695. Clock prescalers for digital filters . . . 2245
Figure 696. Example of one pulse mode . . . 2245
Figure 697. Retriggerable one pulse mode . . . 2247
Figure 698. Measuring time interval between edges on 2 signals . . . 2248
Figure 699. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 2248
Figure 700. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 2249
Figure 701. Control circuit in reset mode . . . 2250
Figure 702. Control circuit in gated mode . . . 2251
Figure 703. Control circuit in trigger mode . . . 2251
Figure 704. TIM15 block diagram . . . 2288
Figure 705. TIM16/TIM17 block diagram . . . 2289
Figure 706. Counter timing diagram with prescaler division change from 1 to 2 . . . 2295
Figure 707. Counter timing diagram with prescaler division change from 1 to 4 . . . 2296
Figure 708. Counter timing diagram, internal clock divided by 1 . . . 2297
Figure 709. Counter timing diagram, internal clock divided by 2 . . . 2298
Figure 710. Counter timing diagram, internal clock divided by 4 . . . 2298
Figure 711. Counter timing diagram, internal clock divided by N . . . 2299
Figure 712. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded) . . . 2299
Figure 713. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded) . . . 2300
Figure 714. Counter timing diagram, internal clock divided by 1 . . . 2301
Figure 715. Counter timing diagram, internal clock divided by 2 . . . 2302
Figure 716. Counter timing diagram, internal clock divided by 4 . . . 2302
Figure 717. Counter timing diagram, internal clock divided by N . . . 2303
Figure 718. Counter timing diagram, update event when repetition counter is not used . . . 2303
Figure 719. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6 . . . 2305
Figure 720. Counter timing diagram, internal clock divided by 2 . . . 2305
Figure 721. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36 . . . 2306
Figure 722. Counter timing diagram, internal clock divided by N . . . 2306
Figure 723. Counter timing diagram, update event with ARPE = 1 (counter underflow) . . . 2307
Figure 724. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 2308
Figure 725. Update rate examples depending on mode and TIM_RCR register settings . . . 2309
Figure 726. Control circuit in normal mode, internal clock divided by 1 . . . 2310
Figure 727. tim_ti2 external clock connection example . . . 2311
Figure 728. Control circuit in external clock mode 1 . . . 2312
Figure 729. Capture/compare channel (example: channel 1 input stage) . . . 2312
Figure 730. Capture/compare channel 1 main circuit . . . 2313
Figure 731. Output stage of capture/compare channel (channel 1) . . . 2313
Figure 732. Output stage of capture/compare channel (channel 2 for TIM15) . . . 2314
Figure 733. PWM input mode timing . . . 2316
Figure 734. Output compare mode, toggle on tim_oc1 . . . 2318

Figure 735. Edge-aligned PWM waveforms (ARR = 8) . . . 2319
Figure 736. Center-aligned PWM waveforms (ARR = 8) . . . 2320
Figure 737. Dithering principle . . . 2321
Figure 738. Data format and register coding in dithering mode . . . 2322
Figure 739. PWM resolution vs frequency . . . 2323
Figure 740. PWM dithering pattern . . . 2323
Figure 741. Dithering effect on duty cycle in center-aligned PWM mode . . . 2324
Figure 742. Generation of a phase-shifted PWM signal with 50% duty cycle . . . 2325
Figure 743. Combined PWM mode on channel 1 and 2 . . . 2327
Figure 744. Complementary output with symmetrical dead-time insertion. . . . 2328
Figure 745. Asymmetrical deadtime . . . 2329
Figure 746. Dead-time waveforms with delay greater than the negative pulse. . . . 2329
Figure 747. Dead-time waveforms with delay greater than the positive pulse. . . . 2329
Figure 748. Clock prescalers for dead-time and digital filters . . . 2330
Figure 749. Break circuitry overview . . . 2332
Figure 750. Output behavior in response to a break event on tim_brk . . . 2334
Figure 751. Immediate break and Delayed 1/2 break modes . . . 2335
Figure 752. Pulse termination during delayed break . . . 2336
Figure 753. Delayed break with deadtime insertion . . . 2336
Figure 754. Output redirection . . . 2338
Figure 755. tim_ocref_clr input selection multiplexer . . . 2339
Figure 756. Asymmetric PWM mode 1 . . . 2340
Figure 757. Asymmetric PWM mode 2 . . . 2340
Figure 758. Combined PWM mode 1 . . . 2341
Figure 759. 6-step generation, COM example (OSSR = 1) . . . 2342
Figure 760. Example of one pulse mode . . . 2343
Figure 761. Retriggerable one pulse mode . . . 2345
Figure 762. Example of counter operation in encoder interface mode. . . . 2347
Figure 763. Example of encoder interface mode with tim_ti1fp1 polarity inverted. . . . 2347
Figure 764. Quadrature encoder counting modes . . . 2348
Figure 765. Direction plus clock encoder mode . . . 2349
Figure 766. Directional clock encoder mode (CC1P = CC2P = 0) . . . 2349
Figure 767. Directional clock encoder mode (CC1P = CC2P = 1) . . . 2350
Figure 768. State diagram for quadrature encoded signals . . . 2351
Figure 769. Encoder mode change with preload transferred on update . . . 2351
Figure 770. Encoder with built-in debouncer, x2 mode . . . 2353
Figure 771. Encoder with built-in debouncer, x4 mode . . . 2353
Figure 772. Clock drift measure using directional clock mode . . . 2354
Figure 773. Measuring time interval between edges on 2 signals . . . 2355
Figure 774. XOR gate connection with TI1S = 1 and XORPS = 0 . . . 2355
Figure 775. XOR gate connection with TI1S = 1 and XORPS = 1 . . . 2356
Figure 776. Control circuit in reset mode . . . 2357
Figure 777. Control circuit in gated mode . . . 2358
Figure 778. Control circuit in trigger mode . . . 2359
Figure 779. LPTIM1 block diagram(1) . . . 2438
Figure 780. Glitch filter timing diagram . . . 2442
Figure 781. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 2443
Figure 782. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 2444
Figure 783. LPTIM output waveform, Continuous counting mode configuration . . . 2444
Figure 784. Waveform generation . . . 2446

Figure 785. Encoder mode counting sequence . . . 2450
Figure 786. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1). . . . 2451
Figure 787. Capture/compare input stage (channel 1) . . . 2452
Figure 788. Capture/compare output stage (channel 1) . . . 2452
Figure 789. Edge-aligned PWM mode (PRELOAD = 1) . . . 2454
Figure 790. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 2455
Figure 791. PWM mode with immediate update versus preloaded update . . . 2456
Figure 792. Independent watchdog block diagram . . . 2482
Figure 793. Reset timing due to timeout . . . 2484
Figure 794. Reset timing due to refresh in the not allowed area . . . 2485
Figure 795. Changing PR, RL, and performing a refresh(1) . . . 2486
Figure 796. Window comparator update(1) . . . 2487
Figure 797. Independent watchdog interrupt timing diagram . . . 2489
Figure 798. Early wake-up comparator update(1) . . . 2490
Figure 799. Watchdog block diagram . . . 2498
Figure 800. Window watchdog timing diagram . . . 2500
Figure 801. RTC block diagram . . . 2505
Figure 802. TAMP block diagram . . . 2559
Figure 803. Backup register protection zones . . . 2566
Figure 804. Tamper sampling with precharge pulse . . . 2571
Figure 805. Low level detection with precharge and filtering . . . 2572
Figure 806. Active tamper filtering . . . 2574
Figure 807. Block diagram . . . 2606
Figure 808. I²C-bus protocol . . . 2608
Figure 809. Setup and hold timings . . . 2610
Figure 810. I2C initialization flow . . . 2612
Figure 811. Data reception . . . 2613
Figure 812. Data transmission . . . 2614
Figure 813. Target initialization flow . . . 2617
Figure 814. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 2619
Figure 815. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 2620
Figure 816. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 2621
Figure 817. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 2622
Figure 818. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 2623
Figure 819. Transfer bus diagrams for I2C target receiver (mandatory events only) . . . 2623
Figure 820. Controller clock generation . . . 2625
Figure 821. Controller initialization flow . . . 2627
Figure 822. 10-bit address read access with HEAD10R = 0 . . . 2627
Figure 823. 10-bit address read access with HEAD10R = 1 . . . 2628
Figure 824. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 2629
Figure 825. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 2630
Figure 826. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . 2631
Figure 827. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 2633
Figure 828. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 2634
Figure 829. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . 2635
Figure 830. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 2639
Figure 831. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 2642
Figure 832. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 2643

Figure 833. Transfer sequence flow for SMBus target receiver N bytes + PEC 2644
Figure 834. Bus transfer diagrams for SMBus target receiver (SBC = 1) 2645
Figure 835. Bus transfer diagrams for SMBus controller transmitter 2646
Figure 836. Bus transfer diagrams for SMBus controller receiver 2648
Figure 837. I3C block diagram 2674
Figure 838. I3C (primary) controller state and programming sequence diagram. 2679
Figure 839. I3C target state and programing sequence diagram 2684
Figure 840. I3C CCC messages, as controller 2698
Figure 841. I3C broadcast ENTDAA CCC, as controller 2699
Figure 842. I3C broadcast, direct read and direct write RSTACT CCC, as controller 2700
Figure 843. I3C CCC messages, as target 2702
Figure 844. I3C broadcast ENTDAA CCC, as target 2703
Figure 845. I3C broadcast DEFTGTS CCC, as target 2704
Figure 846. I3C broadcast DEFGRPA CCC, as target 2705
Figure 847. I3C private read/write messages, as controller. 2707
Figure 848. I3C private read/write messages, as target 2708
Figure 849. Legacy I2C read/write messages, as controller 2709
Figure 850. IBI transfer, as controller/target 2710
Figure 851. Hot-join request transfer, as controller/target 2712
Figure 852. Controller-role request transfer, as controller/target 2713
Figure 853. C-FIFO management, as controller 2714
Figure 854. TX-FIFO management, as controller 2716
Figure 855. RX-FIFO management, as controller 2718
Figure 856. S-FIFO management, as controller 2721
Figure 857. RX-FIFO management, as target on the I3C bus 2722
Figure 858. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus 2724
Figure 859. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target 2726
Figure 860. USART block diagram 2784
Figure 861. Word length programming 2788
Figure 862. Configurable stop bits 2790
Figure 863. TC/TXE behavior when transmitting 2792
Figure 864. Start bit detection when oversampling by 16 or 8 2793
Figure 865. usart_ker_ck clock divider block diagram 2796
Figure 866. Data sampling when oversampling by 16 2797
Figure 867. Data sampling when oversampling by 8 2798
Figure 868. Mute mode using Idle line detection 2805
Figure 869. Mute mode using address mark detection 2806
Figure 870. Break detection in LIN mode (11-bit break length - LBDL bit is set) 2809
Figure 871. Break detection in LIN mode vs. Framing error detection. 2810
Figure 872. USART example of synchronous master transmission. 2811
Figure 873. USART data clock timing diagram in synchronous master mode (M bits = 00) 2811
Figure 874. USART data clock timing diagram in synchronous master mode (M bits = 01) 2812
Figure 875. USART data clock timing diagram in synchronous slave mode (M bits = 00) 2813
Figure 876. ISO 7816-3 asynchronous protocol 2815
Figure 877. Parity error detection using the 1.5 stop bits 2817
Figure 878. IrDA SIR ENDEC block diagram. 2821
Figure 879. IrDA data modulation (3/16) - normal mode 2821
Figure 880. Transmission using DMA 2823

Figure 881. Reception using DMA . . . 2824
Figure 882. Hardware flow control between two USARTs . . . 2824
Figure 883. RS232 RTS flow control . . . 2825
Figure 884. RS232 CTS flow control . . . 2826
Figure 885. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 2829
Figure 886. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 2829
Figure 887. LPUART block diagram . . . 2873
Figure 888. LPUART word length programming . . . 2876
Figure 889. Configurable stop bits . . . 2878
Figure 890. TC/TXE behavior when transmitting . . . 2880
Figure 891. lpuart_ker_ck clock divider block diagram . . . 2884
Figure 892. Mute mode using Idle line detection . . . 2888
Figure 893. Mute mode using address mark detection . . . 2889
Figure 894. Transmission using DMA . . . 2891
Figure 895. Reception using DMA . . . 2892
Figure 896. Hardware flow control between two LPUARTs . . . 2893
Figure 897. RS232 RTS flow control . . . 2893
Figure 898. RS232 CTS flow control . . . 2894
Figure 899. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 2897
Figure 900. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 2897
Figure 901. SPI/I2S block diagram . . . 2929
Figure 902. Full-duplex single master/ single slave application . . . 2932
Figure 903. Half-duplex single master/ single slave application . . . 2933
Figure 904. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) . . . 2934
Figure 905. Master and three independent slaves connected in star topology . . . 2935
Figure 906. Master and three slaves connected in circular (daisy chain) topology . . . 2937
Figure 907. Multimaster application . . . 2938
Figure 908. Scheme of NSS control logic . . . 2940
Figure 909. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) . . . 2940
Figure 910. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) . . . 2941
Figure 911. Data clock timing diagram . . . 2943
Figure 912. Data alignment when data size is not equal to 8, 16 or 32 bits . . . 2944
Figure 913. TI mode transfer . . . 2953
Figure 914. Optional configurations of the slave behavior when an underrun condition is detected . 2955
Figure 915. Waveform examples . . . 2963
Figure 916. Master I2S Philips protocol waveforms (16/32-bit full accuracy) . . . 2964
Figure 917. I2S Philips standard waveforms . . . 2964
Figure 918. Master MSB-justified 16- or 32-bit full-accuracy length . . . 2965
Figure 919. Master MSB-justified 16- or 24-bit data length . . . 2965
Figure 920. Slave MSB-justified 16-, 24- or 32-bit data length . . . 2966
Figure 921. LSB-justified 16 or 24-bit data length . . . 2966
Figure 922. Master PCM when the frame length is equal the data length . . . 2967
Figure 923. Master PCM standard waveforms (16 or 24-bit data length) . . . 2967
Figure 924. Slave PCM waveforms . . . 2968
Figure 925. Startup sequence, I2S Philips standard, master . . . 2971
Figure 926. Startup sequence, I2S Philips standard, slave . . . 2972
Figure 927. Stop sequence, I2S Philips standard, master . . . 2972
Figure 928. I^2S clock generator architecture . . . 2973

Figure 929. Data Format . . . 2975
Figure 930. Handling of underrun situation . . . 2977
Figure 931. Handling of overrun situation . . . 2978
Figure 932. Frame error detection, with FIXCH = 0 . . . 2979
Figure 933. Frame error detection, with FIXCH = 1 . . . 2979
Figure 934. CAN subsystem . . . 3004
Figure 935. FDCAN block diagram . . . 3006
Figure 936. Bit timing . . . 3008
Figure 937. Transceiver delay measurement . . . 3013
Figure 938. Pin control in bus monitoring mode . . . 3014
Figure 939. Pin control in loop-back mode . . . 3017
Figure 940. CAN error state diagram . . . 3018
Figure 941. Message RAM configuration . . . 3019
Figure 942. Standard message ID filter path . . . 3022
Figure 943. Extended message ID filter path . . . 3023
Figure 944. POSENC block diagram . . . 3070
Figure 945. ENDat3 memory organization . . . 3073
Figure 946. POSTX bit transition summary . . . 3080
Figure 947. POSENC strobe interface . . . 3084
Figure 948. POSENC interrupt summary . . . 3086
Figure 949. USB peripheral block diagram . . . 3109
Figure 950. Packet buffer areas with examples of buffer description table locations . . . 3116
Figure 951. UCPD block diagram . . . 3157
Figure 952. Clock division and timing elements . . . 3159
Figure 953. K-code transmission . . . 3161
Figure 954. Transmit order for various sizes of data . . . 3162
Figure 955. Packet format . . . 3163
Figure 956. Line format of Hard Reset . . . 3163
Figure 957. Line format of Cable Reset . . . 3164
Figure 958. BIST test data frame . . . 3165
Figure 959. BIST Carrier Mode 2 frame . . . 3165
Figure 960. UCPD BMC transmitter architecture . . . 3166
Figure 961. UCPD BMC receiver architecture . . . 3167
Figure 962. Block diagram of debug support infrastructure . . . 3195
Figure 963. Product life cycle states and debug authentication . . . 3199
Figure 964. JTAG TAP state machine . . . 3203
Figure 965. CoreSight™ topology . . . 3224
Figure 966. Trace port interface unit (TPIU) . . . 3303
Figure 967. Embedded cross trigger . . . 3315

1 Documentation conventions

1.1 General information

The STM32G56x/55x/54x/53x series devices have an Arm® Cortex®-M33 core with Arm® TrustZone® technology.

arm

Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

Arm and TrustZone are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

1.2 List of abbreviations for registers

The following abbreviations[a] are used in register descriptions:

read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing to the register. The value written to this bit is not important.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to 0. Writing this bit has no effect on the bit value.
read/set by read (rs_r)	Software can read this bit. Reading this bit automatically sets it to 1. Writing this bit has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing 0 has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.

a. This is an exhaustive list of all abbreviations applicable to STMicroelectronics microcontrollers, some of them may not be used in the current document.

read-only write trigger (rt_w1)	Software can read this bit. Writing 1 triggers an event but has no effect on the bit value.
Reserved (Res.)	Reserved bit, must be kept at reset value.

1.3 Register reset value

Bits (binary notation) or bit nibbles (hexadecimal notation) of which the reset value is undefined are marked as X.

Bits (binary notation) or bit nibbles (hexadecimal notation) of which the reset value is unmodified are marked as U.

1.4 Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **AHB**: advanced high-performance bus.

1.5 Availability of peripherals

For detailed availability of peripherals and their number across all sales types (depending on product line and package), refer to the device datasheet.

2 Memory and bus architecture

2.1 System architecture

The architecture relies on a Arm Cortex-M33 core, optimized for execution, as a result of an instruction cache having a direct access to the embedded flash memory.

This architecture also features a 32-bit multilayer AHB bus matrix that interconnects:

- Up to 10 masters
  - Fast C-bus, connecting Cortex-M33 with TrustZone mainline and FPU core C-bus to the internal SRAMs through the instruction cache
  - Slow C-bus, connecting Cortex-M33 with TrustZone mainline and FPU core C-bus to the external memories through the instruction cache
  - Cortex-M33 with TrustZone mainline and FPU core S-bus (three masters connected to three internal SRAMs without latency)
  - Cortex-M33 with TrustZone mainline and FPU core S-bus connected to the external memories through the data cache
  - GPDMA1 (general purpose DMA featuring two master ports)
  - GPDMA2 (general purpose DMA featuring two master ports)
- Up to 9 slaves
  - Internal flash memory
  - Internal SRAM1
  - Internal SRAM2
  - Internal SRAM3
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - XSPI
  - AHB3 peripherals, including AHB to APB bridge and APB peripherals (connected to APB3)
  - AHB4 peripherals

This architecture also features a 32-bit multilayer AHB bus matrix with the interconnections detailed in *Table 1* and *Table 2*.

Table 1. Implementation of masters

Master	Comments	STM32G55x/56x	STM32G53x/54x
Cortex-M33 Fast C-bus	Connecting Cortex-M33 (with Arm TrustZone mainline and FPU) to the internal SRAMs and flash memory through ICACHE	X	X
Cortex-M33 Slow C-bus	Connecting Cortex-M33 (with Arm TrustZone mainline and FPU) to the external memories through ICACHE	X	X
Cortex-M33 S-bus	Connecting the Cortex-M33 (with Arm TrustZone mainline and FPU) to internal SRAMs without latency	3 masters	2 masters
	Connecting Cortex-M33 (with Arm TrustZone mainline and FPU) to the external memories through DCACHE	X	-

Table 1. Implementation of masters (continued)

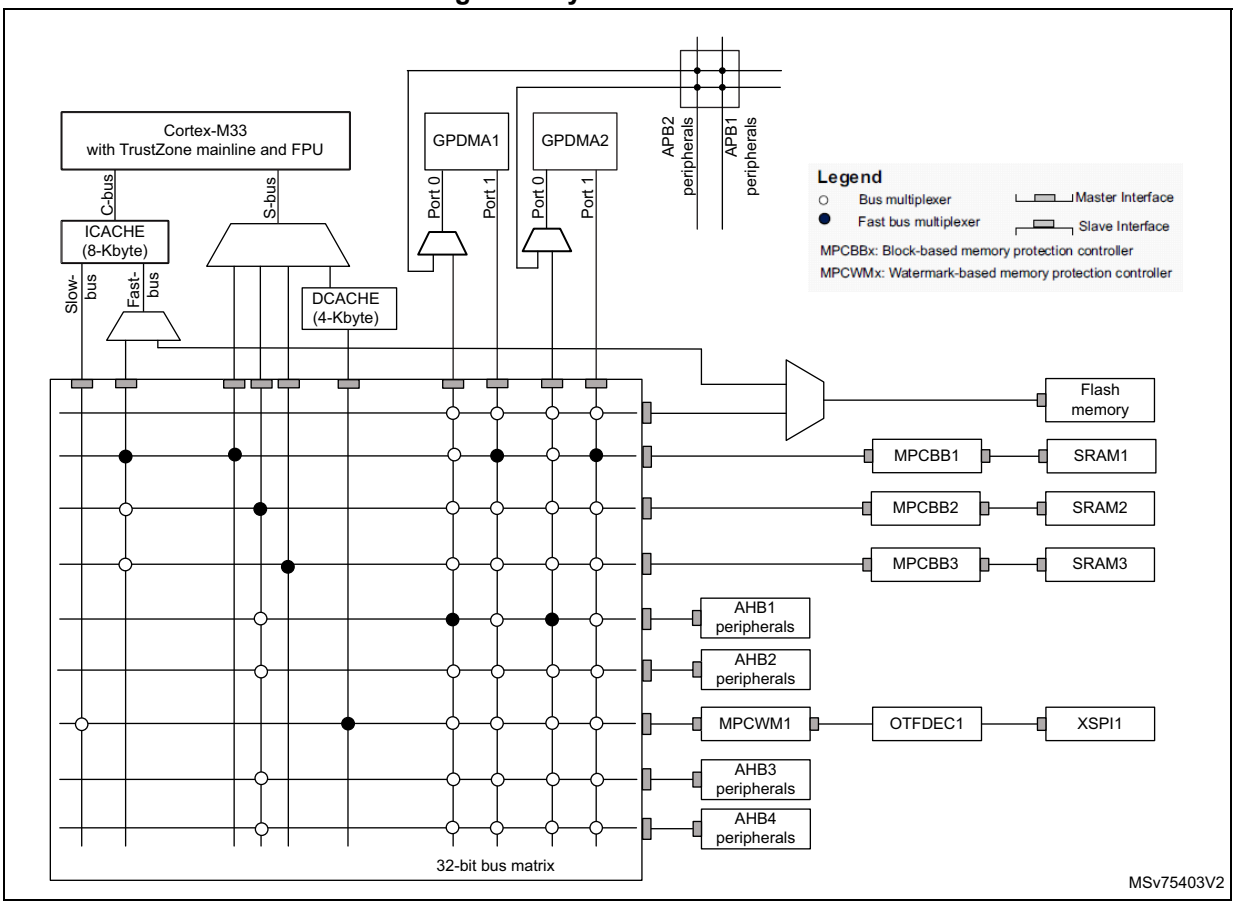
Master	Comments	STM32G55x/56x	STM32G53x/54x
GPDMA1	-	2 masters	
GPDMA2	-	2 masters	

Table 2. Implementation of slaves

Slave	Comments	STM32G55x/56x	STM32G53x/54x
Flash memory	-	X	X
SRAM1	-	X	X
SRAM2	-	X	X
SRAM3	-	X	-
AHB1	Peripherals including AHB to APB bridge, and APB peripherals (connected to APB1 and APB2)	X	X
AHB2	Peripherals	X	X
XSPI1	-	X	-
AHB3	Including AHB to APB bridge, and APB peripherals (connected to APB3)	X	X
AHB4	Peripherals	X	X

The bus matrix provides access from a master to a slave, enabling concurrent accesses and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in *Figure 1*.

Figure 1. System architecture



2.1.1 Fast C-bus

This bus connects the C-bus of the Cortex-M33 core to the internal flash memory and to the BusMatrix via the instruction cache. This bus is used for instruction fetch and data access to the internal memories mapped in code region. This bus targets the internal flash memory and the internal SRAMs (SRAM1, SRAM2, and SRAM3).

SRAM1, SRAM2, and SRAM3 are accessible on this bus with a continuous mapping.

Code execution from SRAM1 code area is always without wait states due to fast bus multiplexer from C-bus to SRAM1. Code execution from SRAM2 or SRAM3 adds 1 wait state per one 32-bit access due to connection through standard bus multiplexer.

2.1.2 Slow C-bus

This bus connects the C-bus of the Cortex-M33 core to the BusMatrix via the instruction cache. This bus is used for instruction fetch and data access to the external memories mapped in code region. This bus targets the external memories (XSPI).

2.1.3 S-bus

This bus connects the system bus of the Cortex-M33 core to the BusMatrix. This bus is used by the core to access data located in a peripheral or SRAM area. This bus targets the

internal SRAMs (SRAM1, SRAM2, and SRAM3), the AHB1 peripherals including the APB1, APB2, AHB2, AHB3, and AHB4 peripherals.

SRAM1, SRAM2, and SRAM3 are accessible on this bus with a continuous mapping.

Code execution is possible also through S-bus but with a penalty that 1 wait state is added per one access.

Note: The Bus Matrix has a zero latency when accessing data from SRAM1, SRAM2, and SRAM3. Code execution through S-bus always add 1 wait state per one access.

2.1.4 DCACHE S-bus

This bus connects the system bus of the Cortex-M33 core to the BusMatrix via the data cache. This bus is used for instruction fetch and data access to the external memories mapped in data region. This bus targets the external memories (XSPI).

Note: Fetching instructions through this bus is less efficient than fetching instructions through the slow C-bus.

2.1.5 GPDMA1 and GPDMA2 buses

These buses connect the four AHB master interfaces of the GPDMA1 and GPDMA2 to the BusMatrix. These buses target the internal flash memory, the internal SRAMs (SRAM1, SRAM2, and SRAM3), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, the AHB3 peripherals, AHB4 peripherals and the external memories through XSPI.

2.1.6 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a round-robin algorithm. This BusMatrix features a fast bus multiplexer used to connect each master to a given slave without latency (see *Figure 1: System architecture*). For the same master, other slaves undergo a latency of at least one cycle at each new access (connection through standard bus multiplexer).

2.1.7 AHB/APB bridges

The three AHB/APB bridges provide full synchronous connections between the AHB and the APB buses, allowing flexible selection of the peripheral frequency.

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to these bridges.

After each device reset, all peripheral clocks are disabled (except for the internal SRAMs and flash memory interfaces). Before using a peripheral, its clock must be enabled in the RCC_AHBxENR and RCC_APBxENR registers.

Note: When an 8- or 16-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 8- or 16-bit data to feed the 32-bit vector.

2.2 TrustZone security architecture

The security architecture is based on Arm TrustZone with the Armv8-M mainline extension.

The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.

When the TrustZone is enabled, the SAU (security-attribution unit) and IDAU (implementation-defined-attribution unit) defines the access permissions based on secure and nonsecure states.

- SAU: Up to eight SAU configurable regions are available for security attribution.
- IDAU: provides a first memory partition as nonsecure or nonsecure callable attributes. The IDAU memory map partition is not configurable, it is fixed by hardware implementation (refer to *Figure 2: Memory map based on IDAU mapping (STM32G55x/56x devices)*). It is then combined with the results from the SAU security attribution, and the higher security state is selected.

Based on IDAU security attribution, the flash memory, system SRAMs and peripheral memory spaces are aliased twice for secure and nonsecure states. However, the external memory space is not aliased.

2.2.1 Default TrustZone security state

When the TrustZone security is activated by the TZEN option bit in the FLASH_OPTR, the default system security state is detailed below:

- CPU:
  - Cortex-M33 is in secure state after reset. The boot address must be at a secure address.
- Memory map:
  - SAU is fully secure after reset. Consequently, all memory map is fully secure. Up to eight SAU configurable regions are available for security attribution.
- Flash memory:
  - The flash memory security area is defined by watermark user options.
  - Block-based security attributions are nonsecure after reset.
- SRAMs:
  - All SRAMs are secure after reset. MPCBBx (block-based memory protection controller) are secure.
- External memories:
  - XSPI banks are secure after reset. MPCWMx (watermark-based memory protection controller) are secure.
- Peripherals (see *Table 3: Securable peripherals by TZSC* and *Table 4: TrustZone-aware peripherals for a list of securable and TrustZone-aware peripherals)*
  - Securable peripherals are nonsecure after reset.
  - TrustZone-aware peripherals are nonsecure after reset. Their secure configuration registers are secure.
- All GPIOs are secure after reset.
- Interrupts:
  - NVIC: All interrupts are secure after reset. NVIC is banked for secure and Nonsecure state.
  - TZIC: All illegal access interrupts are disabled after reset (see *GTZC TrustZone system architecture*).

2.2.2 TrustZone peripheral classification

When the TrustZone security is active, a peripheral can be either securable or TrustZone-aware type as follows:

- Securable: peripheral protected by an AHB/APB firewall gate that is controlled from TZSC controller to define security properties
- TrustZone-aware: peripheral connected directly to AHB or APB bus and implementing a specific TrustZone behavior such as a subset of registers being secure.

Refer to *GTZC TrustZone system architecture* for more details.

The following tables list the securable and TrustZone-aware peripherals within the system.

Table 3. Securable peripherals by TZSC

Bus	Peripheral
AHB4	XSPI1
	DLYB
	HASH
	CCB
	RNG
	SAES
	AES
	PKA
AHB2	DAC1
	DAC2
	DAC5
	DAC6
	DAC7
	DAC8
	DAC9
	DAC10
	ADC1
	ADC2
	ADC3
	FMAC2
AHB1	DCACHE registers
	ICACHE registers
	RAMCFG
	FMAC1
	CORDIC1
	CRC

Table 3. Securable peripherals by TZSC (continued)

Bus	Peripheral
APB3	VREFBUF
	LPTIM1
	TIM20
	TIM26
	TIM27
	COMP1
	COMP2
APB3 (cont'd)	COMP3
	COMP4
	COMP5
	COMP6
	COMP7
	COMP8
	OPAMP1
	OPAMP2
	OPAMP3
	I2C4
	I2C3
	LPUART
	POSENC1
	POSENC2
APB2	HRTIM1
	TIM17
	TIM16
	TIM15
	USART1
	TIM8
	SPI1
	TIM5
	TIM4
	TIM1
	USB

Table 3. Securable peripherals by TZSC (continued)

Bus	Peripheral
APB1	FDCAN1
	FDCAN2
	FDCAN3
	DTS
	CRS
	I2C2
	I2C1
	UART5
	UART4
	UART6
	USART3
	USART2
APB1 (cont'd)	SPI3
	SPI2
	WWDG
	IWDG
	TIM14
	TIM13
	TIM12
	TIM11
	TIM10
	TIM9
	TIM7
	TIM6
	TIM3
	TIM2
	UCPD1
	I3C1

Table 4. TrustZone-aware peripherals

Bus	Peripheral
AHB3	EXTI
	RCC
	PWR

Table 4. TrustZone-aware peripherals (continued)

Bus	Peripheral
AHB4	OTFDEC1
	GPIOH
	GPIOG
	GPIOF
	GPIOE
	GPIOD
	GPIOC
	GPIOB
	GPIOA
AHB1	GTZC
	DCACHE
	ICACHE
	FLASH
	GPDMA2
	GPDMA1
APB3	TAMP
	RTC
	SBS
APB2	PLAY1

2.3 Memory organization

2.3.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

2.3.2 Memory map and register boundary addresses

Figure 2. Memory map based on IDAU mapping (STM32G55x/56x devices)

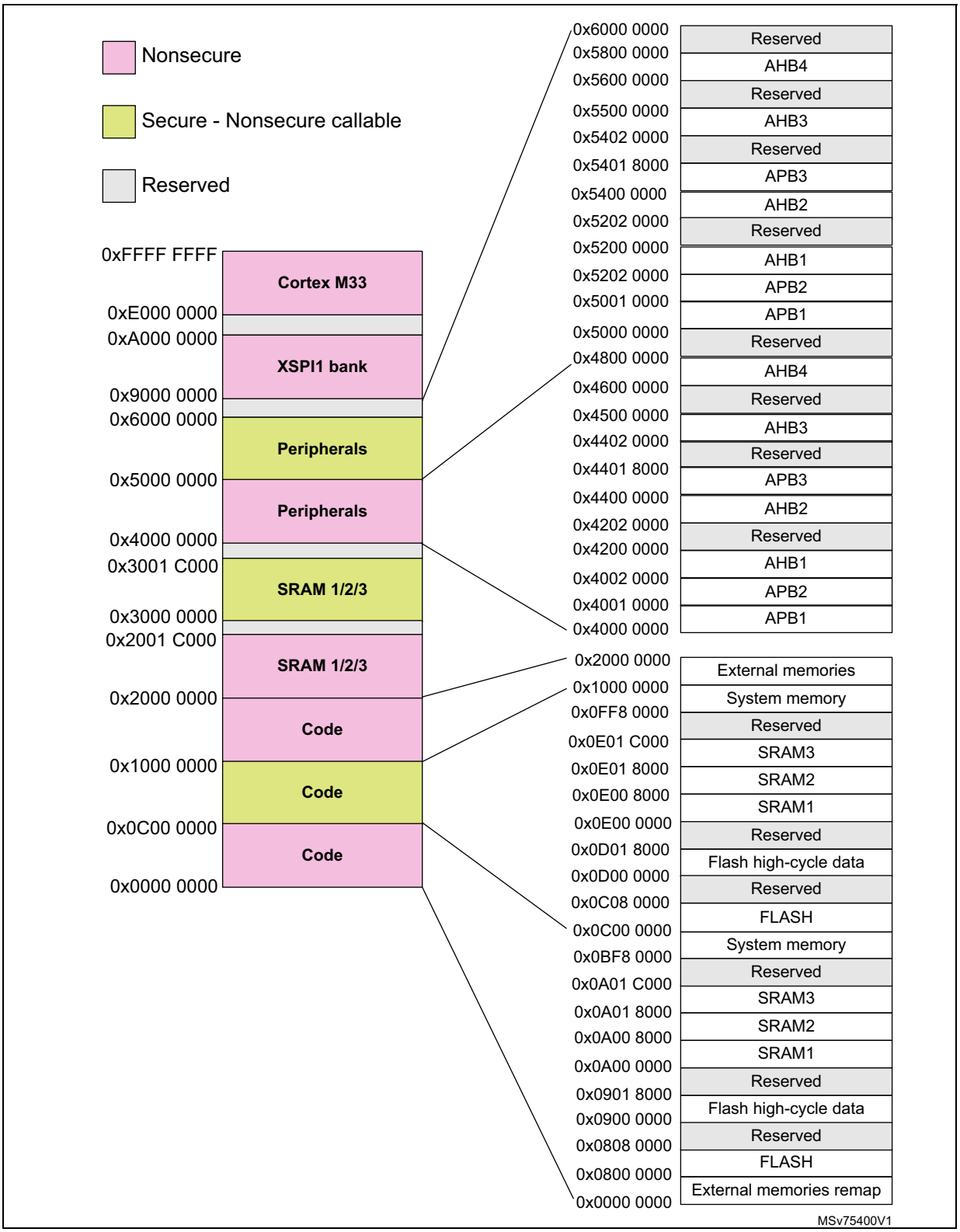


Figure 3. Memory map based on IDAU mapping (STM32G53x/54x devices)

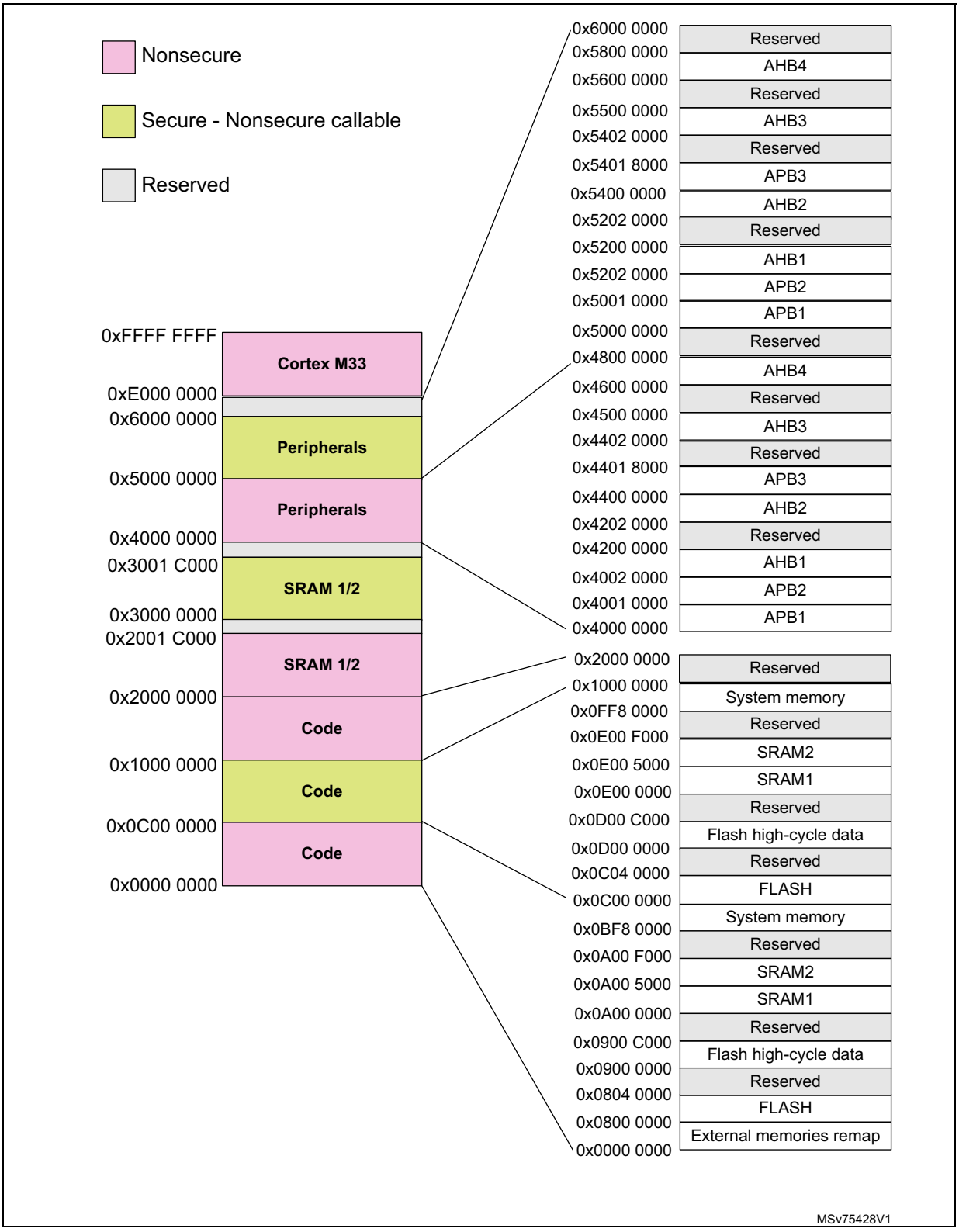


Table 5. Memory map and peripheral register addresses

Bus	Secure boundary address	Nonsecure boundary address	Peripheral
AHB4	0x5700 1800-0x57FF FFFF	0x4700 1800-0x47FF FFFF	Reserved
	0x5700 1400-0x5700 17FF	0x4700 1400-0x4700 17FF	XSPI1
	0x5600 F400-0x5700 13FF	0x4600 F400-0x4700 13FF	Reserved
	0x5600 F000-0x5600 F3FF	0x4600 F000-0x4600 F3FF	DLYBXS1
	0x5600 E000-0x5600 EFFF	0x4600 E000-0x4600 EFFF	Reserved
	0x5600 DC00-0x5600 DFFF	0x4600 DC00-0x4600 DFFF	CCB
	0x5600 A000-0x5600 DBFF	0x4600 A000-0x4600 DBFF	Reserved
	0x5600 8000-0x5600 9FFF	0x4600 8000-0x4600 9FFF	PKA +RAM
	0x5600 7000-0x5600 7FFF	0x4600 7000-0x4600 7FFF	Reserved
	0x5600 6C00-0x5600 6FFF	0x4600 6C00-0x4600 6FFF	SAES
	0x5600 6800-0x5600 6BFF	0x4600 6800-0x4600 6BFF	RNG
	0x5600 6400-0x5600 67FF	0x4600 6400-0x4600 67FF	HASH
	0x5600 6000-0x5600 63FF	0x4600 6000-0x4600 63FF	AES
	0x5600 5400-0x5600 5FFF	0x4600 5400-0x4600 5FFF	Reserved
	0x5600 5000-0x5600 53FF	0x4600 5000-0x4600 53FF	OTFDEC1
	0x5600 2000-0x5600 4FFF	0x4600 2000-0x4600 4FFF	Reserved
	0x5600 1C00-0x5600 1FFF	0x4600 1C00-0x4600 1FFF	GPIOH
	0x5600 1800-0x5600 1BFF	0x4600 1800-0x4600 1BFF	GPIOG
	0x5600 1400-0x5600 17FF	0x4600 1400-0x4600 17FF	GPIOF
	0x5600 1000-0x5600 13FF	0x4600 1000-0x4600 13FF	GPIOE
	0x5600 0C00-0x5600 0FFF	0x4600 0C00-0x4600 0FFF	GPIOD
	0x5600 0800-0x5600 0BFF	0x4600 0800-0x4600 0BFF	GPIOC
	0x5600 0400-0x5600 07FF	0x4600 0400-0x4600 07FF	GPIOB
	0x5600 0000-0x5600 03FF	0x4600 0000-0x4600 03FF	GPIOA
AHB3	0x5402 6400-0x55FF FFFF	0x4402 6400-0x45FF FFFF	Reserved
	0x5402 6000-0x5402 63FF	0x4402 6000-0x4402 63FF	DBGMCU
	0x5402 2400-0x5402 5FFF	0x4402 2400-0x4402 5FFF	Reserved
	0x5402 2000-0x5402 23FF	0x4402 2000-0x4402 23FF	EXTI
	0x5402 1000-0x5402 1FFF	0x4402 1000-0x4402 1FFF	Reserved
	0x5402 0C00-0x5402 0FFF	0x4402 0C00-0x4402 0FFF	RCC
	0x5402 0800-0x5402 0BFF	0x4402 0800-0x4402 0BFF	PWR
	0x5402 0000-0x5402 07FF	0x4402 0000-0x4402 07FF	Reserved

Table 5. Memory map and peripheral register addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Peripheral
APB3	0x5401 8000-0x5401 FFFF	0x4401 8000-0x4401 FFFF	Reserved
	0x5401 4000-0x5401 7FFF	0x4401 4000-0x4401 7FFF	POSENC2
	0x5401 0000-0x5401 3FFF	0x4401 0000-0x4401 3FFF	POSENC1
	0x5400 A400-0x5400 FFFF	0x5400 A400-0x4400 FFFF	Reserved
	0x5400 A000-0x5400 A3FF	0x4400 A000-0x4400 A3FF	COMP8
	0x5400 9C00-0x5400 9FFF	0x4400 9C00-0x4400 9FFF	COMP7
	0x5400 9800-0x5400 9BFF	0x4400 9800-0x4400 9BFF	COMP6
	0x5400 9400-0x5400 97FF	0x4400 9400-0x4400 97FF	COMP5
	0x5400 9000-0x5400 93FF	0x4400 9000-0x4400 93FF	COMP4
	0x5400 8C00-0x5400 8FFF	0x4400 8C00-0x4400 8FFF	COMP3
	0x5400 8800-0x5400 8BFF	0x4400 8800-0x4400 8BFF	COMP2
	0x5400 8400-0x5400 87FF	0x4400 8400-0x4400 87FF	COMP1
	0x5400 8000-0x5400 83FF	0x4400 8000-0x4400 83FF	Reserved
	0x5400 7C00-0x5400 7FFF	0x4400 7C00-0x4400 7FFF	TAMP
	0x5400 7800-0x5400 7BFF	0x4400 7800-0x4400 7BFF	RTC
	0x5400 7400-0x5400 77FF	0x4400 7400-0x4400 77FF	VREFBUF
	0x5400 6800-0x5400 73FF	0x4400 6800-0x4400 73FF	Reserved
	0x5400 6400-0x5400 67FF	0x4400 6400-0x4400 67FF	OPAMP3
	0x5400 6000-0x5400 63FF	0x4400 6000-0x4400 63FF	OPAMP2
	0x5400 5C00-0x5400 5FFF	0x4400 5C00-0x4400 5FFF	OPAMP1
	0x5400 4800-0x5400 5BFF	0x4400 4800-0x4400 5BFF	Reserved
	0x5400 4400-0x5400 47FF	0x4400 4400-0x4400 47FF	LPTIM1
	0x5400 3000-0x5400 43FF	0x4400 3000-0x4400 43FF	Reserved
	0x5400 2C00-0x5400 2FFF	0x4400 2C00-0x4400 2FFF	I2C4
	0x5400 2800-0x5400 2BFF	0x4400 2800-0x4400 2BFF	I2C3
	0x5400 2400-0x5400 27FF	0x4400 2400-0x4400 27FF	LPUART1
	0x5400 1400-0x5400 23FF	0x4400 1400-0x4400 23FF	Reserved
	0x5400 1000-0x5400 13FF	0x4400 1000-0x4400 13FF	TIM27
	0x5400 0C00-0x5400 0FFF	0x4400 0C00-0x4400 0FFF	TIM26
	0x5400 0800-0x5400 0BFF	0x4400 0800-0x4400 0BFF	TIM20
	0x5400 0400-0x5400 07FF	0x4400 0400-0x4400 07FF	SBS
	0x5400 0000-0x5400 03FF	0x4400 0000-0x4400 03FF	Reserved

Table 5. Memory map and peripheral register addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Peripheral
AHB2	0x5202 4800-0x53FF FFFF	0x4202 4800-0x43FF FFFF	Reserved
	0x5202 4400-0x5202 47FF	0x4202 4400-0x4202 47FF	FMAC2
	0x5202 3C00-0x5202 43FF	0x4202 3C00-0x4202 43FF	Reserved
	0x5202 3800-0x5202 3BFF	0x4202 3800-0x4202 3BFF	DAC10
	0x5202 3400-0x5202 37FF	0x4202 3400-0x4202 37FF	DAC9
	0x5202 3000-0x5202 33FF	0x4202 3000-0x4202 33FF	DAC8
	0x5202 2C00-0x5202 2FFF	0x4202 2C00-0x4202 2FFF	DAC7
	0x5202 2800-0x5202 2BFF	0x4202 2800-0x4202 2BFF	DAC6
	0x5202 2400-0x5202 27FF	0x4202 2400-0x4202 27FF	DAC5
	0x5202 1C00-0x5202 23FF	0x4202 1C00-0x4202 23FF	Reserved
	0x5202 1800-0x5202 1BFF	0x4202 1800-0x4202 1BFF	DAC2
	0x5202 1400-0x5202 17FF	0x4202 1400-0x4202 17FF	DAC1
	0x5202 0800-0x5202 13FF	0x4202 0800-0x4202 13FF	Reserved
	0x5202 0400-0x5202 07FF	0x4202 0400-0x4202 07FF	ADC3
	0x5202 0000-0x5202 03FF	0x4202 0000-0x4202 03FF	ADC1/ADC2
AHB1	0x5003 3800-0x5201 FFFF	0x4003 3800-0x4201 FFFF	Reserved
	0x5003 3400-0x5003 37FF	0x4003 3400-0x4003 37FF	MPCBB3
	0x5003 3000-0x5003 33FF	0x4003 3000-0x4003 33FF	MPCBB2
	0x5003 2C00-0x5003 2FFF	0x4003 2C00-0x4003 2FFF	MPCBB1
	0x5003 2800-0x5003 2BFF	0x4003 2800-0x4003 2BFF	GTZC1_TZIC
	0x5003 2400-0x5003 27FF	0x4003 2400-0x4003 27FF	GTZC1_TZSC
	0x5003 1800-0x5003 23FF	0x4003 1800-0x4003 23FF	Reserved
	0x5003 1400-0x5003 17FF	0x4003 1400-0x4003 17FF	DCACHE
	0x5003 0800-0x5003 13FF	0x4003 0800-0x4003 13FF	Reserved
	0x5003 0400-0x5003 07FF	0x4003 0400-0x4003 07FF	ICACHE
	0x5002 7000-0x5003 03FF	0x4002 7000-0x4003 03FF	Reserved
	0x5002 6000-0x5002 6FFF	0x4002 6000-0x4002 6FFF	RAMCFG
	0x5002 4000-0x5002 5FFF	0x4002 4000-0x4002 5FFF	Reserved
	0x5002 3C00-0x5002 3FFF	0x4002 3C00-0x4002 3FFF	FMAC1
	0x5002 3800-0x5002 3BFF	0x4002 3800-0x4002 3BFF	CORDIC1
	0x5002 3400-0x5002 37FF	0x4002 3400-0x4002 37FF	Reserved
	0x5002 3000-0x5002 33FF	0x4002 3000-0x4002 33FF	CRC
	0x5002 2400-0x5002 2FFF	0x5002 2400-0x4002 2FFF	Reserved

Table 5. Memory map and peripheral register addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Peripheral
AHB1	0x5002 2000-0x5002 23FF	0x4002 2000-0x4002 23FF	FLASH registers
	0x5002 1000-0x5002 1FFF	0x4002 1000-0x4002 1FFF	DMA2
	0x5002 0000-0x5002 0FFF	0x4002 0000-0x4002 0FFF	DMA1
APB2	0x5001 7C00-0x5001 FFFF	0x4001 7C00-0x4001 FFFF	Reserved
	0x5001 7800-0x5001 7BFF	0x4001 7800-0x4001 7BFF	TIM5
	0x5001 7400-0x5001 77FF	0x4001 7400-0x4001 77FF	TIM4
	0x5001 6C00-0x5001 73FF	0x4001 6C00-0x4001 73FF	Reserved
	0x5001 6400-0x5001 6BFF	0x4001 6400-0x4001 6BFF	USBSRAM
	0x5001 6000-0x5001 63FF	0x4001 6000-0x4001 63FF	USB
	0x5001 4C00-0x5001 5FFF	0x4001 4C00-0x4001 5FFF	Reserved
	0x5001 4800-0x5001 4BFF	0x4001 4800-0x4001 4BFF	TIM17
	0x5001 4400-0x5001 47FF	0x4001 4400-0x4001 47FF	TIM16
	0x5001 4000-0x5001 43FF	0x4001 4000-0x4001 43FF	TIM15
	0x5001 3C00-0x5001 3FFF	0x4001 3C00-0x4001 3FFF	Reserved
	0x5001 3800-0x5001 3BFF	0x4001 3800-0x4001 3BFF	USART1
	0x5001 3400-0x5001 37FF	0x4001 3400-0x4001 37FF	TIM8
	0x5001 3000-0x5001 33FF	0x4001 3000-0x4001 33FF	SPI1
	0x5001 2C00-0x5001 2FFF	0x4001 2C00-0x4001 2FFF	TIM1
	0x5001 1800-0x5001 2BFF	0x4001 1800-0x4001 2BFF	Reserved
	0x5001 1400-0x5001 17FF	0x4001 1400-0x4001 17FF	PLAY1
	0x5001 1000-0x5001 13FF	0x4001 1000-0x4001 13FF	Reserved
	0x5001 0000-0x5001 0FFF	0x4001 0000-0x4001 0FFF	HRTIM
APB1	0x5000 E000-0x5000 FFFF	0x4000 E000-0x4000 FFFF	Reserved
	0x5000 DC00-0x5000 DFFF	0x4000 DC00-0x4000 DFFF	UCPD1
	0x5000 B800-0x5000 DBFF	0x4000 B800-0x4000 DBFF	Reserved
	0x5000 AC00-0x5000 B7FF	0x4000 AC00-0x4000 B7FF	FDCANSRAM
	0x5000 A800-0x5000 ABFF	0x4000 A800-0x4000 ABFF	FDCAN3
	0x5000 A400-0x5000 A7FF	0x4000 A400-0x4000 A7FF	FDCAN2
	0x5000 A000-0x5000 A3FF	0x4000 A000-0x4000 A3FF	FDCAN1
	0x5000 9000-0x5000 9FFF	0x4000 9000-0x4000 9FFF	Reserved
	0x5000 8C00-0x5000 8FFF	0x4000 8C00-0x4000 8FFF	DTS
	0x5000 6800-0x5000 8BFF	0x4000 6800-0x4000 8BFF	Reserved
	0x5000 6400-0x5000 67FF	0x4000 6400-0x4000 67FF	UART6

Table 5. Memory map and peripheral register addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Peripheral
APB1	0x5000 6000-0x5000 63FF	0x4000 6000-0x4000 63FF	CRS
	0x5000 5C00-0x5000 5FFF	0x4000 5C00-0x4000 5FFF	I3C1
	0x5000 5800-0x5000 5BFF	0x4000 5800-0x4000 5BFF	I2C2
	0x5000 5400-0x5000 57FF	0x4000 5400-0x4000 57FF	I2C1
	0x5000 5000-0x5000 53FF	0x4000 5000-0x4000 53FF	UART5
	0x5000 4C00-0x5000 4FFF	0x4000 4C00-0x4000 4FFF	UART4
	0x5000 4800-0x5000 4BFF	0x4000 4800-0x4000 4BFF	USART3
	0x5000 4400-0x5000 47FF	0x4000 4400-0x4000 47FF	USART2
	0x5000 4000-0x5000 43FF	0x4000 4000-0x4000 43FF	Reserved
	0x5000 3C00-0x5000 3FFF	0x4000 3C00-0x4000 3FFF	SPI3
	0x5000 3800-0x5000 3BFF	0x4000 3800-0x4000 3BFF	SPI2
	0x5000 3400-0x5000 37FF	0x4000 3400-0x4000 37FF	IWDG
	0x5000 3000-0x5000 33FF	0x4000 3000-0x4000 33FF	WWDG
	0x5000 2C00-0x5000 2FFF	0x4000 2C00-0x4000 2FFF	TIM14
	0x5000 2800-0x5000 2BFF	0x4000 2800-0x4000 2BFF	TIM13
	0x5000 2400-0x5000 27FF	0x4000 2400-0x4000 27FF	TIM12
	0x5000 2000-0x5000 23FF	0x4000 2000-0x4000 23FF	TIM11
	0x5000 1C00-0x5000 1FFF	0x4000 1C00-0x4000 1FFF	TIM10
	0x5000 1800-0x5000 1BFF	0x4000 1800-0x4000 1BFF	TIM9
	0x5000 1400-0x5000 17FF	0x4000 1400-0x4000 17FF	TIM7
	0x5000 1000-0x5000 13FF	0x4000 1000-0x4000 13FF	TIM6
	0x5000 0800-0x5000 0FFF	0x4000 0800-0x4000 0FFF	Reserved
	0x5000 0400-0x5000 07FF	0x4000 0400-0x4000 07FF	TIM3
	0x5000 0000-0x5000 03FF	0x4000 0000-0x4000 03FF	TIM2

2.3.3 Embedded SRAM

STM32G55x/56x devices feature up to 112-Kbyte SRAMs:

- 32-Kbyte SRAM1
- 64-Kbyte SRAM2
- 16-Kbyte SRAM3

STM32G53x/54x devices feature up to 60-Kbyte SRAMs:

- 20-Kbyte SRAM1
- 40-Kbyte SRAM2

These SRAMs can be accessed as bytes, half-words (16 bits), or full words (32 bits). These memories can be addressed both by CPU and DMAs.

The CPU can access the SRAM1, SRAM2, and SRAM3 through the system bus or through the C-bus depending on the selected address.

When the TrustZone® security is enabled, all SRAMs are secure after reset. The SRAM can be programmed as nonsecure with a block granularity. For more details, refer to *Section 5: Global TrustZone® controller (GTZC)*.

SRAM features are detailed in *Section 6.3.1: Internal SRAMs features*.

2.3.4 Flash memory overview

The flash memory is composed of two distinct physical areas:

- The main flash memory block that contains the application program and user data (dual bank).
- The information block that is composed of the following parts:
  - Option bytes for hardware and memory protection user configuration
  - System memory that contains ST proprietary code
  - OTP (one-time programmable) area

The flash interface implements instruction access and data access based on the AHB protocol. It also implements the logic necessary to carry out the flash memory operations (program/erase) controlled through the flash registers plus security access control features. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

2.3.5 Boot modes

At startup, the core jumps to the boot address configured through the BOOT0 pin, the BOOT_ADDR option bytes, the TZEN option bit, debug request, and the product state.

If BOOT_ADDR is not yet configured, dedicated libraries (programmed during ST production), can be used for secure boot. They are located in system flash memory:

- ST libraries in system flash memory assist the application software boot with special features such as secure boot and secure firmware install (SFI-RSS).
- ST iROT (immutable root of trust) secure software in user flash memory is used for secure firmware update and provisioning (SFU).

If a debugger is attached to the product, the entry point is a debug authentication policy, which is used to unlock the device when attached to a debugger. A digital signature must be provided to perform a regression to product state, where debug is allowed.

Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. Refer to the application note STM32 microcontroller system memory boot mode (AN2606).

3 System security

The devices are designed with a comprehensive set of security features, some of them based on the standard Arm® TrustZone® technology.

This section introduces the purpose these features serve in fulfilling security functions. Describing how the security functions are addressed is essential in evaluating IoT devices against security standards and regulations such as CRA. They also significantly reduce the cost and complexity of software development for OEM and third-party developers by facilitating the reuse, improving the interoperability, and minimizing the API fragmentation.

This section lists the different security functions and how to achieve them using features available on the devices.

3.1 Key security features

Product security is built on secure HW features:

- Secure isolation and privilege mode based on Armv8-M architecture of the Cortex-M33 core, extended to securable I/O, memories, and peripherals.
- Temporal isolation separates boot levels using HDPL (hide protect level) monotonic counter.
- Life cycle scheme based on PRODUCT_STATE transitions
  - Debug Authentication allows product maintenance in-the-field (debug reopening or regression)

  - Allows protected firmware distribution in up to three steps: immutable root of trust, secure (TZ-Closed), and nonsecure application (Closed/Locked)
- Cryptographic acceleration
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM, and CCM chaining modes
  - Secure AES 256-bit security coprocessor, supporting ECB, CBC, CTR, GCM, and CCM chaining modes with side-channel counter measures and mitigations
  - HASH processor supporting SHA-1 checksum, SHA-2 secure hash (224, 256, 384 and 512) and six Keccak-based functions (SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, and SHAKE256)
  - Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel counter measures and mitigations when manipulating secrets
  - True random number generator (RNG), NIST SP800-90B pre-certified
- On-the-fly decryption of encrypted image stored on external flash memory
  - Low or zero latency with standard NOR flash memories
  - Can be used to encrypt the image using the device unique secret keys
  - Automatic key-erase in case of tamper
  - Supported on STM32G56x
- Active tamper signals and protection against temperature, voltage, and frequency attacks
  - Up to two active inputs, two active output tamper pins, available in different power modes
- Identifiers preventing replay and cloning, such as UID, EPOCH and PUF (HUK).

Following root security services (RSS) are provided by the code in the product system memory, only fully supported on STM32G56x and partially on STM32G54x:

- RSSSF guarantees Secure firmware installation (SFI) with device unique cryptographic key pair
- RSSiSB is the entry point for secure boot controlling integrity of a user flash code. It is typically used with STiRoT.
- RSSREG providing minimum services to serve Debug Authentication, allowing control for PRODUCT_STATE regression.

Following secure firmware is proposed for installation in user flash memory:

- STiRoT for Secure Boot and Secure Update extending STiSB.
- STDA for Debug Authentication control extension.
- STiReg provides services involving certificate management.
- STuRoT as an updatable root of trust extension.

3.2 Secure install

*T*he secure firmware install (SFI) is a STMicroelectronics secure service authenticated and decrypted by the immutable RSS code stored in the device. The SFI allows secure and counted installation of OEM firmware in an untrusted production environment (such as OEM contract manufacturer).
The confidentiality of the installed images written in the internal flash memory or encrypted in an external flash memory, is also protected, using the AES.

The SFI native service is an immutable Root Secure Service (RSS) implemented in the system flash area of the nonvolatile memory. Leveraging the on-chip immutable bootloader supports the image download through USART, USB, SPI, FDCAN, I2C, and JTAG.
SFI is supported by STM32G56x and not implemented on STM32G54x/53x/52x because it relies on secure HW cryptography (refer to *Section 35: Secure AES coprocessor (SAES)*).
Further information can be found in AN4992 *"Overview secure firmware install (SFI)"*.

3.3 Secure boot

Secure boot relies on immutable code that is always executed after a system reset. As a root of trust, this code checks the device static protections and activates available device runtime protections, reducing the risk of invalid or malicious code running on the platform. As root of trust, the secure boot also checks the integrity and authenticity of the next level firmware before executing it.
The product always boots from CPU1 for all configurations.
On STM32G56x/54x devices, there are two options using BOOT_UBE (refer to *Section 14.3.6: SBS boot control*) to execute code after a reset:

- Booting through security services RSSiSB when selected. RSSiSB controls integrity of a user flash code. The user flash code can be a ST-iROT (provided by ST) or an OEM code.
- Booting directly in user flash when the OEM manages the full chain (OEM-iROT). This is the default configuration.

On STM32G55x/53x devices, the secure boot must be implemented in user flash. Secure boot starting point in user flash is fixed using a combination of configurable boot address and BOOT_LOCK option, that becomes immutable (refer to *Section 7.4.7: Description of boot address option bytes*).
The secure boot typically initializes the secure storage and installs on-the-fly decryption keys in the OTFDEC (STM32G56x), to be able to use encrypted firmware stored in an external flash memory.
The secure boot shall take advantage of hardware security features:

- Unique boot entry
- Resource isolation using TrustZone (refer to *Section 5: Global TrustZone® controller (GTZC)*)
- Boot modes (refer to *Section 4: Boot modes*)
- SBS boot control (HDPL) (refer to *Section 14.3.6: SBS boot control*)
- HDPL storage (refer to *Section 7.6.2: Hide protection (HDP)*)

3.3.1 Unique boot entry

Using the unique boot entry (BOOT_UBE), it is possible to select the boot entry point between security services in system flash (RSSiSB) and proprietary boot entry (OEMiROT) located in user flash memory. This selection is possible only for products embedding cryptographic acceleration when TrustZone is enabled (TZEN = 0xB4).
When TrustZone is enabled and SBOOT_LOCK secure option bit is cleared, the application selects a boot entry point located either in the system flash memory (refer to *Section 3.3.2*), or in the secure user flash memory, at the address defined by SBOOTADD option bytes.
When TrustZone is enabled and SBOOT_LOCK secure option bit is set, the device unique

boot entry is the unmodifiable secure address defined by SBOOTADD option bytes. The application cannot modify these option bytes when SBOOT_LOCK is set.
When TrustZone is disabled (TZEN = 0xC3) and BOOT_LOCK option bit is cleared, the application selects a boot entry point located in the nonsecure user flash memory, at the address defined by BOOTADD option bytes.
When TrustZone is disabled and BOOT_LOCK option bit is set, the device unique boot entry is the unmodifiable nonsecure address defined by BOOTADD option bytes. The application cannot modify these option bytes when BOOT_LOCK is set.

Note: SBOOT_LOCK and BOOT_LOCK can be changed only when PRODUCT_STATE = Open, Provisioning, or Regression.

BOOT_UBE is not applicable on products not supporting hardware crypto accelerators.

For more information on the boot mechanisms, refer to *Section 4: Boot modes*.

3.3.2 Immutable root of trust in system flash memory

The immutable root-of-trust code (RSSFS) stored in the system flash memory is first used to load and verify extensions (ST extensions: RSS-e-XXX). One typical extension is the RSS-e-SFI that allows SFI for secure and counted installation of OEM firmware in untrusted production environments (such as an OEM contract manufacturer).
The immutable code also includes secure runtime services that can be called at runtime, when a secure application sets the SBS_RSSCMDR register to a non-null value, before triggering a system reset. This runtime feature is deactivated when the BOOT_LOCK secure option bit is set or when the PRODUCT_STATE is different from Open.

3.3.3 Temporal isolation levels

The hardware and software resources used during boot can be isolated. This is called temporal isolation, based on a monotonic counter, with respect to the incrementation of the levels. When the counter is incremented, the resources (code and data) of the previous levels are hidden.

This increases security because it is impossible to execute the boot process partially from within the application, and thus compromise security.

For more details refer to (*Section 14.3.6: SBS boot control*).

System flash functions JumpHDPLvlx are available in RSSLib to securely facilitate the HDPL increment.

3.4 Secure update

The secure firmware update is a secure service that runs after a secure boot. Its actual functions depend on the availability of the TrustZone features, and on the firmware stored in the device.

Cortex-M is commonly used to run trusted boot and a trusted OS to create a Trusted Execution Environment (TEE). The trusted boot and TEE application, supported by the STM32 ecosystem, allow the update of the microcontroller built-in program with new firmware versions, adding new features and correcting potential issues. The update process is performed in a secure way, to prevent unauthorized updates and access to confidential on-device data.

A firmware update can be done either on a single firmware image including both secure and nonsecure parts, or on the secure (respectively nonsecure) part of the firmware image, independently.

In the devices, the secure update application leverages the same hardware security as the firmware install described in *Section 3.2*.

3.4.1 Live update

STM32G5 series includes an option to minimize the downtime during transition from one code version to another. In dual bank device, a new code is loaded to the other bank, while the present version is still fully functional. Refer to *Section 3.4.1: Live update*.

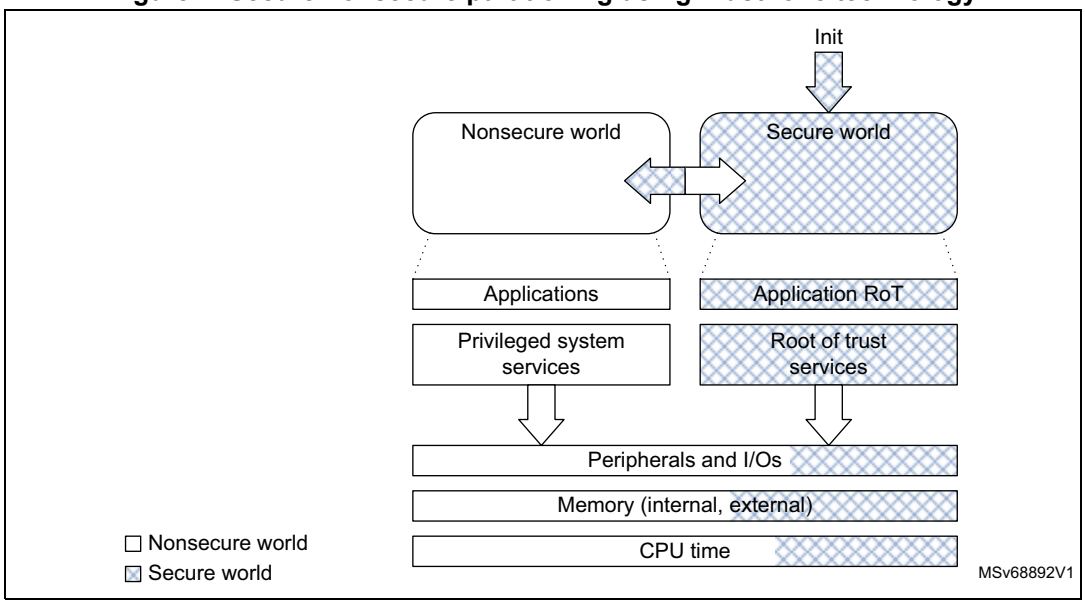
There are two ways to swap the user flash banks and switch to the new content. Either the device reboot with new bank mapping, or the banks are swapped live during execution of a small portion of code that must be left unmodified. Refer to *Section 7.3.12: Flash bank swapping*.

Security of swap operations must be guaranteed by execution protection and authentication of the code image. If the security conditions are not met, or the swapping is not desirable, it can be securely disabled, both temporarily and permanently.

3.5 Resource isolation using TrustZone

The hardware and software resources can be partitioned so that they exist in the secure or in the nonsecure world, as shown in the figure below.

Figure 4. Secure/nonsecure partitioning using TrustZone technology



Note: The initial partitioning of the platform is under the responsibility of the secure firmware executed after device reset.

Thanks to this resource isolation technology, the secure world can be used to protect critical code against intentional or unintentional tampering from the more exposed code running in the nonsecure world.

Note: The secure code is typically small and rarely modified, while the nonsecure code is more exposed, and prone to firmware updates.

3.5.1 TrustZone security architecture

The Armv8-M TrustZone technology is a comprehensive hardware architecture that proposes to developers a comprehensive, holistic protection across the entire processor and system. The device TrustZone hardware features include:

- The Armv8-M mainline security extension of Cortex-M33, enabling a new processor secure state, with its associated secure interrupts
- The dynamic allocation of memory and peripherals to TrustZone using eight security attribution unit (SAU) regions of Cortex-M33
- Global TrustZone framework (GTZC) extending the TrustZone protection against transactions coming from masters in the system different from Cortex-M33
- TrustZone-aware embedded flash memory and peripherals

Note: The TZEN option bit in the FLASH_OPTSR2_PRG register activates TrustZone security.

3.5.2 Armv8-M security extension of Cortex-M33

The Arm security extension of the Cortex-M33 is an evolution, not a revolution. It uses the programmer model from earlier Cortex-M subfamilies like Cortex-M4. Indeed, Armv8-M is architecturally similar to Armv7-M, using the same 32-bit architecture, the same memory mapped resources protected with an MPU. Armv8-M also uses the nested vector interrupt controller (NVIC).

The implementation on STM32G5 devices is composed of the following features:

- A new processor state, with almost no additional code/cycle overhead (as opposed to Armv8-A TrustZone) that uses a dedicated exception routine to trigger a secure/nonsecure world change
- Two memory map views of a shared 4-Gbyte address space
- A low interrupt latency for both secure and nonsecure domains, and a new interrupt configuration for security grouping and priority setting
- Separated exception vector tables for the secure and nonsecure exceptions
- Micro-coded context preservation
- Banking of specific registers across secure/nonsecure states, including stack pointers with stack-limit checkers
- Banking of the following Cortex-M33 programmable components (two separate units for secure and nonsecure):
  - SysTick timer
  - MPU configuration registers (12 MPU regions in secure, eight in nonsecure)
  - Some of the system control block (SCB) registers
- New system exception (SecureFault) for handling of security violations
- Configurable debug support, as defined in *Section 3.8.3*

For more information, refer to the Cortex-M33 programming manual (PM0264).

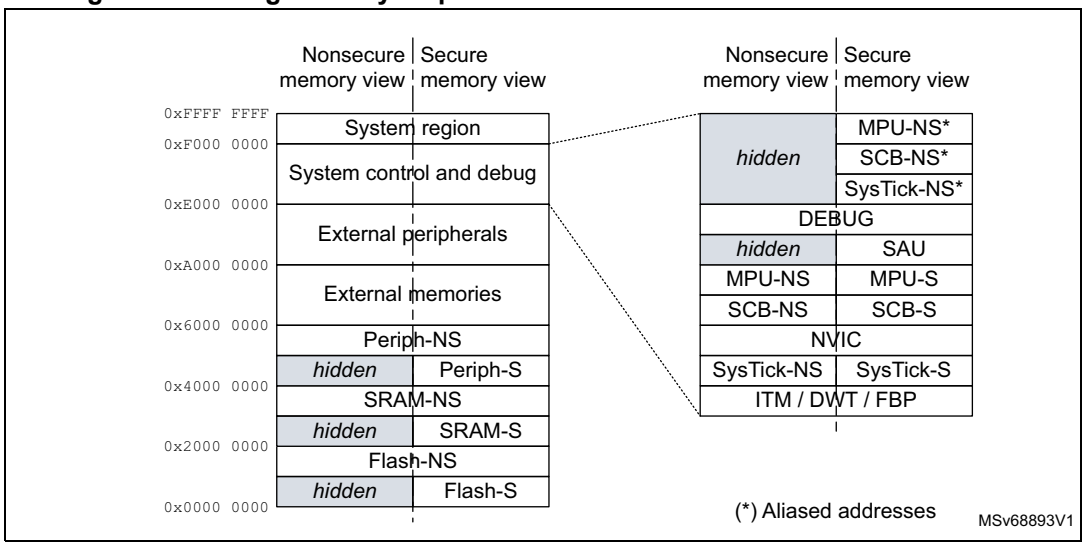
3.5.3 Memory and peripheral allocation using IDAU/SAU

Security attributes

As illustrated in *Figure 5*, the Armv8-M nonsecure memory view is similar to Armv7-M (found in Cortex-M4), with the difference that the secure memory is hidden. The secure memory view shows the flash memory, SRAM and peripherals that are accessible only while the Cortex processor executes in Secure state.

Note: Figure 5 shows the 32-bit address space after SAU configuration, as viewed by the secure code.

Figure 5. Sharing memory map between CPU in secure and nonsecure state



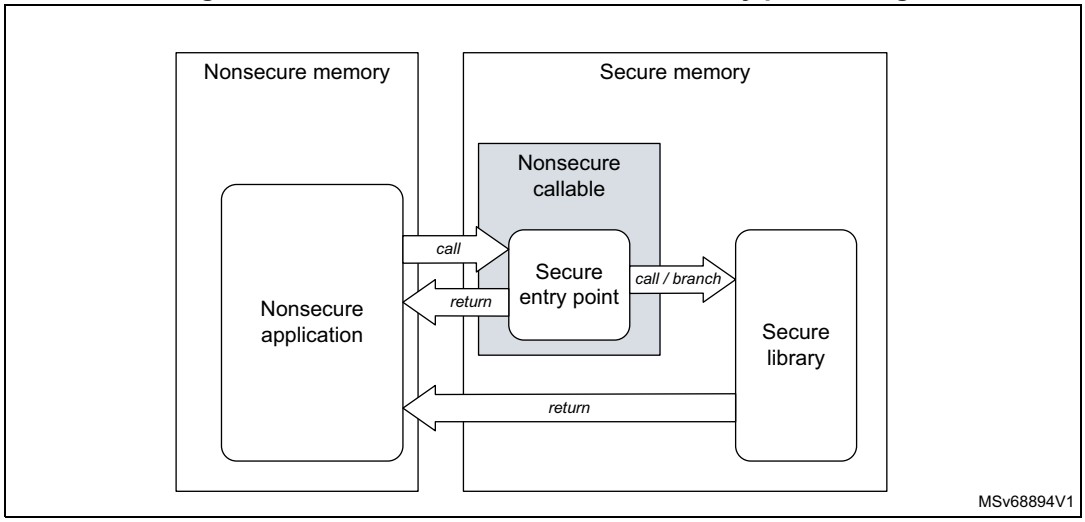
The Cortex processor state (and the associated rights) depends upon the security attribute assigned to the memory region where it is executed:

- A processor in a nonsecure/secure state executes only from nonsecure (NS)/secure (S) program memory.
- When running in secure state, the processor can access data from both S and NS memories. When running in nonsecure state, the CPU is limited to nonsecure memories.

To manage transitions to the secure world, developers must create nonsecure callable (NSC) regions containing valid entry points to the secure libraries. The first instruction in

these entry points must be the new secure gate (SG) instruction, used by the nonsecure code to call a secure function (see the figure below).

Figure 6. Secure world transition and memory partitioning



Programming security attributes

In Cortex-M33, the static implementation defined attribution unit (IDAU) works with the programmable security attribution unit (SAU) to assign a specific security attribute (S, NS, or NSC) to a specific address, as shown in the table below.

Table 6. Configuring security attributes with IDAU and SAU

IDAU security attribution	SAU security attribution(1)	Final security attribution
Nonsecure	Secure	Secure
	Secure-NSC	Secure-NSC
	Nonsecure	Nonsecure
Secure-NSC	Secure	Secure
	Nonsecure	Secure-NSC

1. Defined regions are aligned to 32-byte boundaries.

The Cortex-M33 can configure the SAU only in the secure-privilege state. When the TrustZone is enabled, the SAU defaults all addresses as secure (S). A secure boot application can then program the SAU to create NSC or NS regions, as shown in *Table 6*.

Note: The SAU/IDAU settings are applicable only to the Cortex-M33*. The other masters, like DMA, are not affected by these policies.*

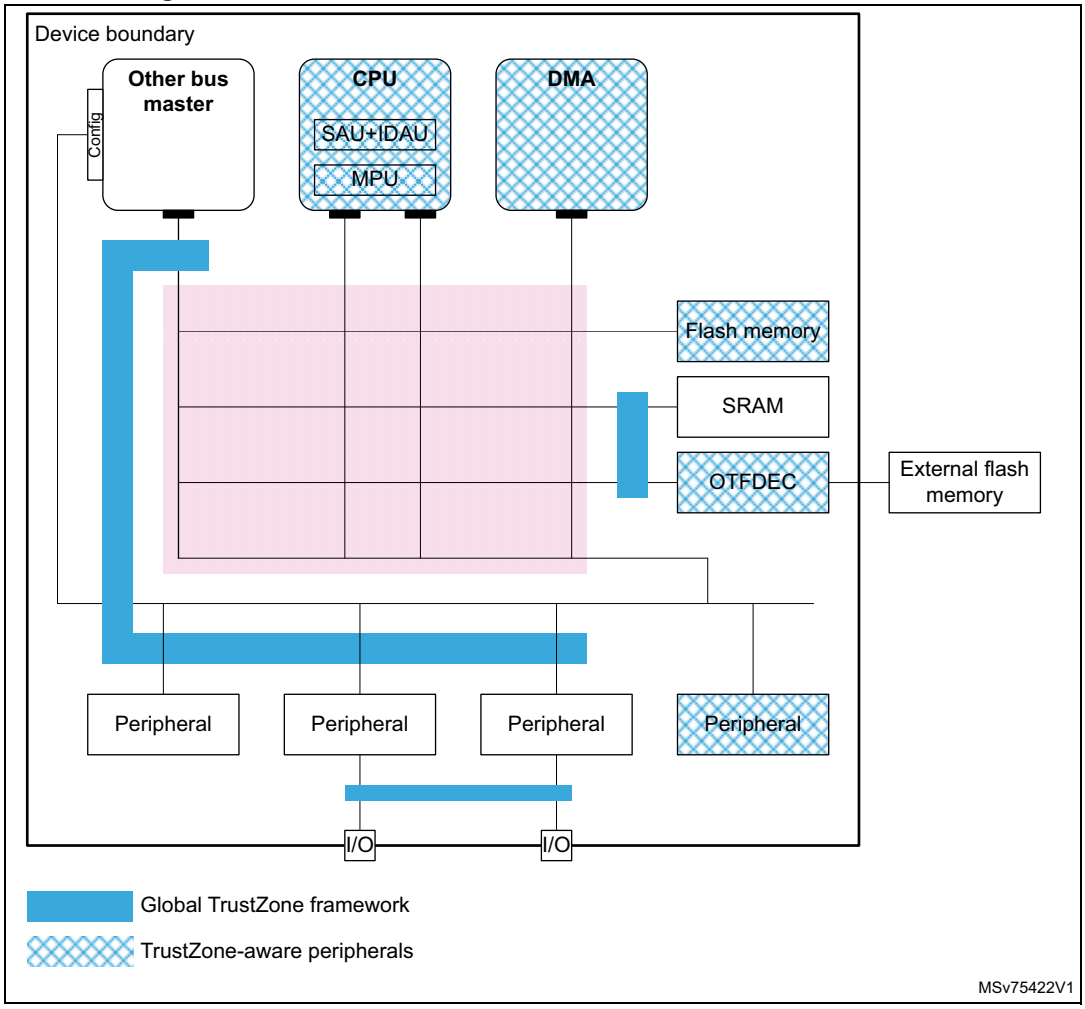
3.5.4 Memory and peripheral allocation using GTZC

Global TrustZone framework architecture

On top of the Armv8-M TrustZone security extension in Cortex-M33, the devices embed complementary security features that reinforce, in a flexible way, the isolation between secure and nonsecure worlds. Unlike the SAU/IDAU, the GTZC can protect legacy

memories and peripherals against nonsecure transactions coming from other masters than the Cortex-M33.

Figure 7. Global TrustZone framework and TrustZone awareness



Securing peripherals with TZSC

When the TrustZone security is active, a peripheral is either securable through the GTZC, or is natively TrustZone-aware, as shown in *Figure 7*:

- A securable peripheral or memory is protected by an AHB/APB firewall gate, controlled by the TrustZone security controller (TZSC).
- A TrustZone-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific TrustZone behavior, such as a subset of secure registers or a secure memory area.

TrustZone-aware AHB masters like Cortex-M33 or DMAs, drive a secure signal in the AHB interconnect, according to their security mode, independently to the GTZC.

Securing memories with TZSC and MPCBB

The TZSC block in GTZC provides the capability to manage the security and privilege for all securable external memories, programming the MPCWM resources as defined in the table below.

Table 7. MPCWMx resources for STM32G55x/56x

Memory	MPC resource	Type of filtering	Number of regions	Default security	On-the-fly decryption(1)
XSPI1	MPCWM1	Nonsecure, privileged, or unprivileged region (watermarks)	2	Secure privileged(2)	Yes

1. Using the OTFDEC.
2. Assuming TrustZone is activated on the device, nonsecure unprivileged otherwise.

The MPCBB resources in GTZC provide the capability to configure the security and privilege of embedded SRAM blocks, as defined in the tables below.

Table 8. MPCBBx resources for STM32G53x/54x

Memory	MPC resource	Type of filtering	Memory size	Block size	Number of super-blocks	Default security
SRAM1	GTZC1_MPCBB1	Block-based, managing security and privilege	20 KB	512(1) bytes	2	Secure privileged(2)
SRAM2	GTZC1_MPCBB2		40 KB		3	

1. Blocks are grouped in super-blocks of 32 consecutive blocks, to manage the configuration locking.
2. Assuming TrustZone is activated on the device, nonsecure unprivileged otherwise.

Table 9. MPCBBx resources for STM32G55x/56x

Memory	MPC resource	Type of filtering	Memory size	Block size	Number of super-blocks	Default security
SRAM1	GTZC1_MPCBB1	Block-based, managing security and privilege	32 KB	512(1) bytes	4	Secure privileged (2)
SRAM2	GTZC1_MPCBB2		64 KB		8	
SRAM3	GTZC1_MPCBB3		16 KB		2	

1. Blocks are grouped in super-blocks of 32 consecutive blocks, to manage the configuration locking.
2. Assuming TrustZone is activated on the device, nonsecure unprivileged otherwise.

Applying GTZC configurations

The TZSC and MPCBB blocks can be used in one of the following ways:

- Statically programmed during the secure boot, locked and not changed afterwards
- Dynamically reprogrammed using a specific application code or real-time kernel

When the dynamic option is selected and the configuration is not locked:

- MPCBB secure blocks or MPCWM nonsecure region size can be changed by a secure software. This software must be privileged for MPCWM, can be unprivileged if the particular block is not privileged-only.
- The secure (respectively privilege) state of each peripheral can be changed writing to GTZC_TZSC_SECCFRGx (respectively GTZC_TZSC_PRIVCFGRx) registers.

Securing peripherals with TZSC

The TZSC block in GTZC provides the capability to manage the security and the privilege for all securable peripherals.

Note: When the TrustZone is deactivated, the resource isolation hardware GTZC can still be used to isolate peripherals to privileged code only.

When the TrustZone is activated, peripherals are set as nonsecure and unprivileged after reset.

TrustZone-aware peripherals

The devices include the following TrustZone-aware peripherals:

- GPIOA to GPIOH
- GTZCx_MPCBB, GTZCx_TZIC and GTZCx_TZSC (GTZC blocks)
- OTFDEC1, writable only in secure if TZEN is enabled
- EXTI
- Flash memory
- RCC and PWR
- GPDMA1 and GPDMA2
- SBS registers
- RTC and TAMP
- MCU debug unit DBGMCU
- PLAY1

TrustZone illegal access controller (TZIC)

The TZIC block in GTZC gathers all illegal access events originated from sources either protected by GTZC or TrustZone-aware, generating one global secure interrupt towards the NVIC.

TZIC is available only when the system is TrustZone enabled (TZEN = 0xB4). All accesses to TZIC registers must be secured.

For each illegal event source, a status flag and a clear bit exist. Each illegal event can be masked, not generating an interrupt toward the NVIC.

Note: By default, all events are masked.

3.5.5 Managing security in TrustZone-aware peripherals

This section gives more details on how the security is implemented in the TrustZone-aware peripherals listed in the previous section.

Embedded flash memory

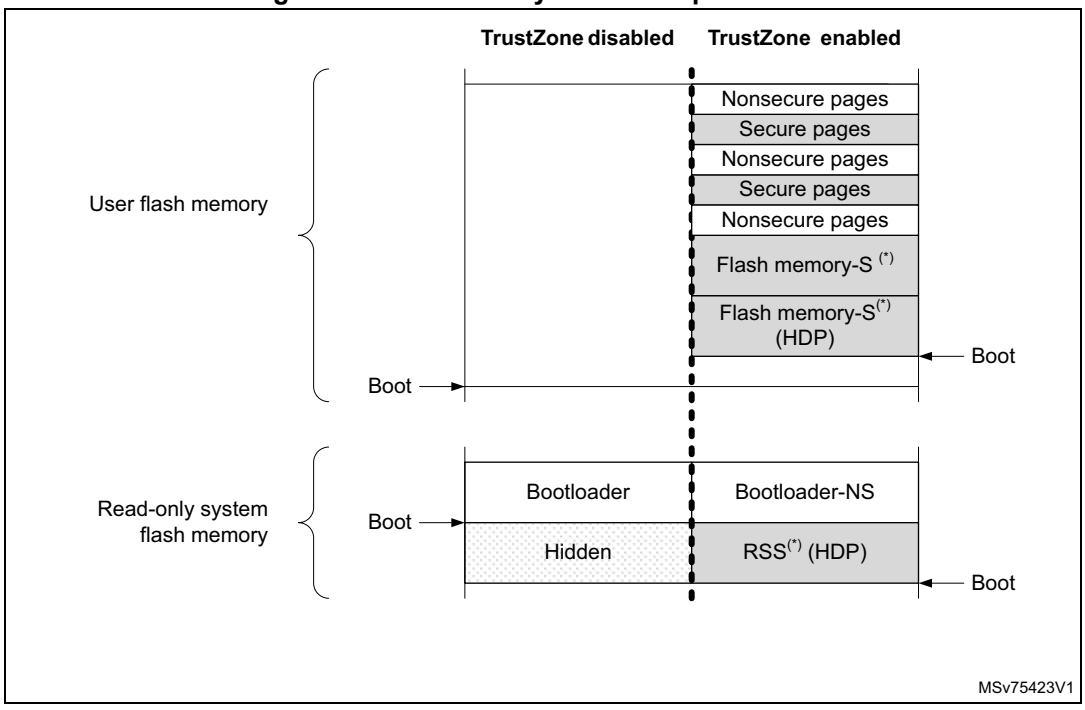
When the TrustZone security is enabled through option bytes (TZEN = 0xB4), the whole memory is secure after reset. The following protections, shown in *Figure 8*, are available to the application:

- nonvolatile user secure areas, defined with nonvolatile secure user option bytes
  - watermark-based secure only area (x2)
  - secure hide protection (HDP) area, stickily hidden after boot (x2)
- volatile user secure pages, defined with volatile secure registers (lost after reset)
  - Any page set as nonsecure (example: outside watermark-based secure only area), can be set as secure on-the-fly using the block-based configuration registers.

Note: All areas are aligned on the page granularity of the flash memory, which can be configured as secure while it is tagged as nonsecure in Cortex-M33 IDAU/SAU. In this case, nonsecure accesses by the CPU to the flash memory are denied.

Erase or program operations can be available to secure/nonsecure code only for secure/ nonsecure pages or memory. A flash memory is considered secure if at least one page is secure.

Figure 8. Flash memory TrustZone protections



As shown above, when TrustZone is enabled (TZEN = 0xB4), the application code can use the HDP area that is part of the flash memory watermark-based secure area. Indeed, when

the application changes the HDPL level, data read, write, and instruction fetch on this HDP area is denied until the next system reset.

For example, the software code in the secure flash memory HDP area can be executed only once, with any further access to this area denied until the next system reset. Additionally, any flash memory page belonging to an active HDP area cannot be erased anymore.

When the TrustZone is disabled (TZEN = 0xC3), the volatile/non-volatile secure area features are deactivated and all secure registers are RAZ/WI.

See *Section 7: Embedded flash memory (FLASH)* for more details.

On-the-fly encryption/decryption (OTFDEC)

When the TrustZone security is enabled (TZEN = 0xB4), the OTFDEC can be initialized only by secure applications. Each of the four encrypted regions, once the configuration is confirmed, can be write-locked until the next power-on-reset.

Note: This resource is only supported by STM32G56x, not implemented on STM32G53x/54x/55x.

Any application (secure or nonsecure) can verify the initialization context of each OTFDEC region (including CRC of the keys), by reading the peripheral registers.

Key registers in each OTFDEC are write-only.

Direct memory access controllers (GPDMAx)

When a DMA channel x is defined as secure (SECx = 1 in GPDMA_SECCFGR), the source and destination transfers can be independently set as secure or nonsecure by a secure application using SSEC and DSEC bits in GPDMA_CxTR1. The table below summarizes these security options available in each DMA channel.

Table 10. DMA channel use (security)(1)

Destination type	Secure DMA channel x (SECx = 1)		Nonsecure DMA channel y (SECy = 0)	
	Secure source	Nonsecure source	Secure source	Nonsecure source
Secure destination	OK	OK(2)	Transfer blocked	
Nonsecure destination	OK(3)	OK(4)	Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return 0s. An illegal access event to TZIC is automatically triggered by the memory/peripheral used as source or destination.
2. If the source is a memory, the transfer is possible only if SSEC = 0, otherwise the transfer is blocked.
3. If the destination is a memory, the transfer is possible only if DSEC = 0, otherwise the transfer is blocked.
4. If the transfer is memory-to-memory, it is possible only if SSEC = 0 and DSEC = 0, otherwise it is blocked.

When a channel is configured as secure:

- Registers allocated to this channel (excluding GPDMA_SECCFGR, GPDMA_PRIVCFGR and GPDMA_RCFGLOCKR) are read as 0. Writes are ignored for nonsecure accesses. A secure illegal access event may also be triggered towards the TZIC peripheral.
  - Writes to GPDMA_SECCFGR and GPDMA_RCFGLOCKR must be secure. For each bit in GPDMA_PRIVCFGR, write must be secure if the corresponding bit in GPDMA_SECCFGR is set.
- In linked-list mode, the loading of the next linked-list data structure from memory is performed with secure transfers.

- When switching to a nonsecure state, the secure application must abort the channel or wait until the secure channel is completed before doing the switch.

Note: DMA secure channels are not available when TrustZone is deactivated. When a channel is configured as nonsecure, in linked-list mode, the loading of the next linked-list data structure from memory is performed with nonsecure transfers.

See *Section 16: General purpose direct memory access controller (GPDMA)* for more details.

Power control (PWR)

When the TrustZone security is enabled (TZEN = 0xB4), the selected PWR registers can be secured through PWR_SECCFGR, protecting the following PWR features:

- Low power mode setup
- Wake-up (WKUP) pins definition
- Voltage detection and monitoring
- RTC domain control

Other PWR configuration bits become secure:

- When the system clock selection is secure in the RCC: the voltage scaling (VOS) becomes secure.
- When a GPIO is configured as secure: its corresponding bit for pull-up/pull-down configuration in Standby mode becomes secure.
- When the USB Type-C/USB power delivery interface (UCPD) is configured as secure in TZSC: PWR_UCPDR register becomes secure.

See *Section 10: Power control (PWR)* for details.

Secure clock and reset (RCC)

When the TrustZone security is enabled (TZEN = 0xB4) and security is enabled in the RCC, the bits controlling the peripheral clocks and resets become TrustZone-aware:

- If the peripheral is securable and programmed as secure in the TZSC, the peripheral clock and reset bits become secure.
- If the peripheral is TrustZone-aware, the peripheral clock and reset bits become secure when at least one function is configured as secure inside the peripheral.

Additionally, the following configurations can be made secure-only using RCC_SECCFGR:

- External clock (such as HSE or LSE), internal oscillator (such as HSI or LSI)
- Main PLL and AHB prescaler
- System clock source selection
- MCO clock output
- Reset flag
- Automatic internal oscillator waking up configuration

See *Section 11: Reset and clock control (RCC)* for details.

Real-time clock (RTC)

Like all TrustZone-aware peripherals, a nonsecure read/write access to a secured RTC register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the RTC illegal access event is enabled in the TZIC.

After a RTC domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes. The secure boot code can then change the security setup, making registers Alarm A, Alarm B, wakeup timer, and timestamp secure or not, using RTC_SECCFGR.

When the SEC bit is set in secure-only RTC_SECCFGR:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER, and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When SEC is cleared in secure-only RTC_SECCFGR, it is still possible to restrict access in secure mode to some RTC registers by setting dedicated control bits: INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC, and ALRBSEC.

Note: The RTC security configuration is not affected by a system reset.

See*Section 49: Real-time clock (RTC)* for more details.

Tamper and backup registers (TAMP)

Like all TrustZone-aware peripherals, a nonsecure read/write access to a secured TAMP register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the TAMP illegal access event is enabled in the TZIC.

After a RTC domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes. The secure boot code can change this security setup, making some registers secure or not as needed, using TAMP_SECCFGR register.

When TAMPSEC is set in TAMP_SECCFGR:

- Writing the TAMP registers is possible only in secure mode. Backup registers have their own write protection (see below).
- Reading the TAMP registers (exception TAMP_SECCFGR, TAMP_PRIVCFGR, and TAMP_MISR) returns 0 if the access is nonsecure. Backup registers have their own read protection (see below).

The application can also:

- make TAMP_COUNTR register read and write secure-only by setting the CNT1SEC bit in TAMP_SECCFGR secure register
- increase security in backup registers for two of the three protection zones configured using BKPRWSEC[7:0] and BKPWSEC[7:0] bitfields in TAMP_SECCFGR:
  - protection zone 1 is read nonsecure, write nonsecure
  - protection zone 2 is read nonsecure, write secure
  - protection zone 3 is read secure, write secure

Note: The TAMP security configuration is not affected by a system reset.

See *Section 50: Tamper and backup registers (TAMP)* for more details.

General-purpose I/Os (GPIO)

When the TrustZone security is enabled (TZEN = 0xB4), each I/O pin of the GPIO ports can be individually configured as secure through the GPIOx_SECCFGR registers. Only a secure application can write to GPIOx_SECCFGR registers. After boot, each I/O pin is set as secure.

When an I/O pin is configured as secure, its corresponding configuration bits for alternate function (AF), mode selection (MODE), and I/O data are RAZ/WI in case of nonsecure access.

When a digital alternate function is used (input/output mode) to protect data transiting from/to the I/O managed by a secure peripheral, the devices add a secure alternate function gate on the path between the peripheral and its allocated I/Os:

- if the peripheral is secure, the I/O pin must also be secure to allow input/output of data
- if the peripheral is not secure, the connection is allowed regardless of the I/O pin state.

The TrustZone-aware logic around GPIO ports, used as alternate functions, is summarized in the table below.

Table 11. Secure alternate function between peripherals and allocated I/Os

Security configuration		Alternate function logic		Comment
Peripheral	**Allocated I/O pin**	**Input**	**Output**	
Secure	Secure	I/O data	Peripheral data	-
Nonsecure				Out of reset configuration
Secure	Nonsecure	0	0	-
Nonsecure		I/O data	Peripheral data	

When an analog function with an analog switch is used, the connection to the peripherals listed in the table below is blocked by hardware when the peripheral is nonsecure and the I/O is secure.

Table 12. Nonsecure peripheral functions that cannot be connected to secure I/Os

Peripheral	Analog function(1)	Input	Output
ADC12	ADC12_INy (y = 1 to 17)	X	-

1. Used to find the I/O corresponding to the signal/function on the package (refer to the product datasheet).

Finally, regarding GPIO and security, *Table 13* summarizes the list of I/Os without any hardware protection linked to TrustZone. The listed signals (input and/or outputs) are not blocked when the I/O is set as secure, and the associated peripheral is nonsecure.

For example, when a secure application sets PA4 as secure to be used as LPTIM2_CH1, if the DAC is nonsecure, it can be programmed to output data to PA4, potentially causing malfunction to the secure application.

Similarly, when a secure application sets PA0 as secure to be used as UART4_TX, if TAMP is nonsecure, it can be programmed to capture the USART input traffic through the TAMP_IN signal.

It is important that, for each case described in *Table 13*, the secure application decides if a potential effect on data integrity or confidentiality is critical or not. For example, if the USART situation described above is not acceptable (data transiting on secure USART is confidential), then the secure application must configure the TAMP as secure even if not used by the secure application.

Table 13. Nonsecure peripheral functions that can be connected to secure I/Os

Peripheral	Signal[1]	Input	Output	How to set the peripheral or function as secure
DAC	DAC1_OUTx (x = 1, 2)	-	X	Set DAC1SEC in GTZC1_TZSC_SECCFGR1.
UCPD	UCPD1_CCx (x = 1, 2)	X	X	Set UCPD1SEC in GTZC1_TZSC_ SECCFGR1.
	UCPD1_DBx (x = 1, 2)	X	-	
TAMP	TAMP_INx (x = 1 to 8)	X	-	Set TAMPSEC in TAMP_SECCFGR
	TAMP_OUTx (x = 1 to 8)	-	X	
RTC	RTC_OUTx (x = 1, 2)	-	X	Set SEC in RTC_SECCFGR.
	RTC_TS	X	-	Set TSSEC in RTC_SECCFGR.
PWR	WKUPx (x = 1 to 8)	X	-	Set WUPxSEC in PWR_SECCFGR.
RCC	LSCO	-	X	Set LSESEC in RCC_SECCFGR.
EXTI	EXTIx (x= 0 to 58)	X	-	Set SECx bit in EXTI_SECCFGR.

1. To find the I/O corresponding to the signal/function on the package, refer to the product datasheet.

Refer to *Section 13: General-purpose I/Os (GPIO)* for more details.

Extended interrupts and event controller (EXTI)

When the TrustZone security is enabled, the EXTI is able to protect event register bits from being modified by nonsecure accesses. The protection can individually be activated per input event via the register bits in EXTI_SECCFGRx. When an input event is configured as secure, only a secure application can change the configuration (including security), change the masking or clear the status of this input event.

The security configuration in EXTI_SECCFGR1 and EXTI_SECCFGR2 can be globally locked after reset in EXTI_LOCKR.

See *Section 19: Extended interrupts and event controller (EXTI)* for more details.

System configuration, boot, and security (SBS)

Like all TrustZone-aware peripherals, when the TrustZone security is enabled, a nonsecure read/write access to a secured SBS register is RAZ/WI. Such access also generates an illegal access event that triggers a secure illegal access interrupt if the SBS illegal access event is not masked in the TZIC.

See *Section 14: System configuration, boot, and security (SBS)* for more details.

Microcontroller debug unit (DBGMCU)

The MCU debug component (DBGMCU) helps the debugger, providing support for:

- low-power modes behavior during debug
- peripheral freeze during debug, applicable to I2Cs, IWDG, WWDG, timers, low-power timers, and GPDMA channels

The DBGCMU is a TrustZone-aware peripheral, managing accesses to its control registers as described in the table below.

Table 14. TrustZone-aware DBGMCU nonsecure accesses management

Debug profile	Peripheral status(1)	DBG_xx_STOP control bits	
		Write access	Read access
Nonsecure invasive (SPIDEN = 0)	NS	Yes (S(2) or NS)	Yes (S or NS)
	S	None (S or NS)	
Secure invasive (SPIDEN =1)	NS	Yes (S or NS)	
	S	Yes (S only)	

1. As reported by the GTZC, the TrustZone-aware peripheral or the DMA channel.
2. Secure access from debugger is converted to nonsecure access in the device.

Refer to *Section 60: Debug support (DBG)* for more details.

3.5.6 Activating TrustZone security

The TrustZone is deactivated by default in all devices. It can be activated by setting the TZEN option bit in FLASH_OPTSR2_PRG when PRODUCT_STATE is Open. Once TZEN has changed from disabled to enabled (0xC3 to 0xB4), the default security state, after reset, is always the following:

- CPU subsystem
  - Cortex-M33 exits reset in secure state, hence the boot address must point toward a secure memory area.
  - All interrupt sources are secure (in NVIC).
  - The memory mapped viewed by the CPU through IDAU/SAU is fully secure.
- Embedded flash memory
  - Flash memory nonvolatile secure areas (with their HDP zone), are defined with nonvolatile registers FLASH_SECWMxR (x = 1, 2). Default secure option bytes setup is all user flash secure, without HDP area defined.
  - Volatile block-based security attributions of the flash memory are nonsecure.
- Embedded SRAMs
  - All SRAMs are secure, as defined in GTZC/MPCBB. The secure boot code can change this security setup, making blocks secure or not.
- External memories
  - All memory devices connected to XSPI1 are secure. The secure-boot code can change this security setup, making components secure or not.
- All GPIOs are secure.
- All GPDMA channels are nonsecure.
- Backup registers are nonsecure.
- Peripherals and GTZC:
  - Securable peripherals are nonsecure and unprivileged.
  - TrustZone-aware peripherals are nonsecure, with their secure configuration registers being secure.

– All illegal access interrupts in GTZC/TZIC are disabled.

3.5.7 Deactivating TrustZone security

Once TrustZone is activated, it can only be deactivated during a PRODUCT_STATE regression to Open.

Note: Such PRODUCT_STATE regression triggers the erasing of embedded memories (SRAM2, flash), and the reset of all peripherals, including the OTFDEC and all crypto engines.

After the TrustZone deactivation, the following features are no longer available:

- The nonvolatile secure area of the embedded flash memory is deactivated, including the HDP area.
- Only the NVIC manages nonsecure interrupts.
- All secure registers in TrustZone-aware peripherals are RAZ/WI.

3.5.8 Resource isolation using Cortex privileged mode

In parallel to the TrustZone isolation described in *Section 3.5*, the hardware and software resources can be partitioned so that they are restricted to software running in Cortex privileged mode.

Thanks to this hardware isolation technology, available even if TrustZone is disabled, critical code or data can be protected against intentional or unintentional tampering from the more exposed unprivileged code.

Memory and peripheral privileged allocation using MPU

The Cortex-M33 MPU divides the unified memory into eight regions in nonsecure and twelve in TrustZone, each aligned to a multiple of 32 bytes. Each memory region can be programmed to generate faults when accessed inappropriately by unprivileged software.

Memory and peripheral privileged allocation using GTZC

For the Cortex-M33 master, to complement the coarse isolation provided by the MPU, the GTZC reinforces, in a flexible way, the isolation between privileged and unprivileged tasks, for peripherals and selected memories.

- Securing peripherals with TZSC (privileged-only)

  In the devices, a peripheral is either securable privileged-only through GTZC, or is natively privileged-aware:

  – A securable privileged-only peripheral or memory is protected by an AHB/APB firewall gate controlled by the TZSC.

  – A privileged-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific behavior (for example, a subset of registers Privilege-aware masters like GPDMA1 and GPDMA2, drive privileged signal in the

AHB interconnect according to their internal privileged mode, independently to the GTZC.

- Securing memories with TZSC and MPCBB (privileged-only)

  The TZSC logic in GTZC provides the capability to manage the privilege level for all securable external memories, programming the MPCWM resources.

  Similarly, the TZSC logic in GTZC provides the capability to configure the privilege level of embedded SRAM blocks, programming the MPCBB resources.

- Error management (privileged-only)
  - Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal unprivileged read and write accesses.
  - The addressed resource follows a silent-fail behavior, returning all-0 data for read and ignoring any write.
  - When an illegal unprivileged access occurs, no bus error is generated, except when this is an instruction fetch, accessing a privileged memory or a peripheral register.

Managing security in privileged-aware peripherals

TrustZone-aware peripherals also implement privileged-only access mode. The privileged protection is valid even if TrustZone is disabled (TZEN = 0xC3):

- Embedded flash memory

  By default, all registers can be read or programmed in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in FLASH_PRIVCFGR, reading and writing the secure registers is possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0xC3).

  When nonsecure privileged bit NSPRIV is set in FLASH_PRIVCFGR, reading and writing the flash memory nonsecure registers is possible only in privileged mode.

  Regarding privileged protection, the devices offer the following features:
  - The system flash memory can be accessed in both privileged and unprivileged modes.
  - Each watermark-based secure area, including its secure HDP area, is accessible in secure-privileged and secure-unprivileged mode, if applicable.
  - Each 8-Kbyte page of the embedded flash memory can be programmed on-the-fly as privileged only, using the block-based privileged configuration registers FLASH_PRIV1BBRx and FLASH_PRIV2BBRx. An unprivileged page is accessible by privileged or unprivileged access.

Note: Switching a page from privileged to unprivileged does not erase its content.

When applicable, an erase or program operation is always available to privileged code, and is available to unprivileged code only for unprivileged pages or unprivileged memory.

- On-the-fly encryption/decryption (OTFDEC)

  When privileged bit PRIV is set in OTFDEC_PRIVCFGR, only a privileged application can initialize the OTFDEC.

Note: OTFDEC_PRIVCFGR can be read by both privileged and unprivileged code.

- Direct memory access controllers (GPDMAx)

  When a DMA channel x is defined as privileged (PRIVx = 1 in GPDMA_PRIVCFGR), special rules (see the table below) apply when accessing privileged/unprivileged source or destination.

Table 15. DMA channel use (privilege)

Destination	Privileged DMA channel x (PRIVx = 1)		Unprivileged DMA channel y (PRIVy = 0)	
	Privileged source	Unprivileged source	Privileged source	Unprivileged source
Privileged	OK		Transfer blocked(1)	
Unprivileged			Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return 0s.

- Power control (PWR)

  By default, after a power-on or a system reset, all PWR registers except PWR_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in PWR_PRIVCFGR, reading and writing the PWR securable registers are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0xC3).

  When nonsecure privileged bit NSPRIV is set in PWR_PRIVCFGR, reading and writing the PWR nonsecure registers are possible only in privileged mode.

  See *Section 10: Power control (PWR)* for details.

- Secure clock and reset (RCC)

  By default, after a power-on or a system reset, all RCC registers except RCC_PRIVCFGR can be read or written in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in RCC_PRIVCFGR, reading and writing the RCC securable bits are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0xC3).

  When nonsecure privileged bit NSPRIV is set in RCC_PRIVCFGR, reading and writing the RCC nonsecure bits are possible only in privileged mode.

  See *Section 11: Reset and clock control (RCC)* for details.

- Real time clock (RTC)

  By default after any RTC domain reset, all RTC registers except RTC_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

  When PRIV bit is set in privileged-only RTC_PRIVCFGR:

  – Writing the RTC registers is possible only in privileged mode.

  – Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privileged and unprivileged modes.

  All the other RTC registers can be read only in privileged mode.

  When PRIV bit is cleared in privileged-only RTC_PRIVCFGR register, it is still possible to restrict access to privileged mode to some RTC registers by setting dedicated control bits: INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV, or ALRBPRIV.

  See *Section 49: Real-time clock (RTC)* for details.

- Tamper and backup registers (TAMP)

  By default after any RTC domain reset, all TAMP registers except TAMP_PRIVCFGR can be read or written in both privileged and unprivileged modes.

  When PRIV bit is set in privileged-only TAMP_PRIVCFGR:

  – Writing the TAMP registers is possible only in privileged mode, except for the backup registers and the monotonic counters that have their own protection setting.

  – Reading the TAMP_SECCFGR or TAMP_PRIVCFGR is always possible in privilege and unprivileged modes. All the other TAMP registers can be read only in

privilege mode, except for the backup registers and the monotonic counters that have their own protection setting.

The application can also:

– make TAMP_COUNT1R register read and write privileged-only by setting the CNTPRIV bit in TAMP_PRIVCFGR

– increase security for two of the three protection zones in backup registers, using BKPRWPRIV and BKPWPRIV bits in TAMP_PRIVCFGR:

– Make protection zone 1 read privileged, write privileged.

– Make protection zone 2 read privileged or unprivileged, write privileged.

– Protection zone 3 is always read and write privileged or unprivileged.

- General-purpose I/Os (GPIO)

  All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or nonsecure).

- Extended interrupts and event controller (EXTI)

  The EXTI peripheral is able to protect event register bits from being modified by unprivileged accesses. The protection is individually activated per input event via the register bits in the privileged-only EXTI_PRIVCFGRx registers. When an input event is configured as privileged, only a privileged application can change the configuration (including security if applicable), change the masking or clear the status of this input event.

  The security configuration in EXTI_PRIVCFGR1 and EXTI_PRIVCFGR2 can be globally locked after reset in EXTI_LOCKR.

  See *Section 19: Extended interrupts and event controller (EXTI)* for more details.

- System configuration boot and security (SBS)

  All SBS registers can be read and written in both privileged and unprivileged modes, except:

  – FPUSEC bit in SBS_SECCFGR registers (privileged only)

  – SBS registers for CPU configuration: SBS_CSLCKR, SBS_FPUIMR, SBS_CNSLCKR, SBS_EPOCHSELCR, SBS_ECCNMIR, SBS_ECCNMIR, SBS_SWAPLOCKR and SBS_FBMODER

  See *Section 14: System configuration, boot, and security (SBS)* for more details.

3.6 Secure execution

Through a mix of special software and hardware features, the devices ensure the correct operation of their functions against abnormal situations caused by programmer errors, software attacks through network access or local attempt for tampering code execution.

This section describes the hardware features specifically designed for secure execution.

3.6.1 Memory protection unit (MPU)

The Cortex-M33 includes a memory protection unit (MPU) that can restrict the read and write accesses to Each memory region (including those mapped to peripherals), based on one or more of the following parameters

- Cortex-M33 operating mode (privileged, unprivileged)
- data/instruction fetch

The memory map and the programming of the nonsecure and secure MPU split memory into regions (up to eight for the nonsecure, and up to twelve for the secure). Secure MPU is only available when TrustZone is activated.

3.6.2 Embedded flash memory write protection

The embedded flash memory write protection (WRP) prevents illegal or unwanted write/erase to special sections of the embedded flash memory user area (system area is permanently write protected).

Write protected area is defined through the option bytes.

Write protected areas can be modified using option bytes only until PRODUCT_STATE = Provisioned.

Note: Bank erase is not possible when it contains a write-protected area (WRP or HDP area).

3.6.3 Tamper detection and response

Principle

The devices include active protection of critical security assets against temperature, voltage and frequency attacks, with the following features:

- erasure of device secrets upon tamper detection
- improved guarantee of safe execution for the CPU and its associated security peripherals, including:
  - out-of-range voltage (example: V_{DD}, V_{CORE}, V_{REF+}), temperature and clocking (LSE, HSE) detection
  - security watchdog IWDG clocked by the internal oscillator LSI
  - possible selection of internal oscillator HSI as system clock
- power supply protection
  - RTC/TAMP domain powered with V_{DD}

See *Section 50: Tamper and backup registers (TAMP)* for more details.

Tamper detection sources

The devices support four active input/output pins, allowing two independent active-tamper meshes, or up to two meshes if the same output pin is shared by several input pins (for a total of two active-tamper I/Os).

The active pins are clocked by the LSE, and are functional in different system operating modes (Run, Sleep, Stop, or Standby). Refer to TAMP pins functionality over modes for list of tamper pins and their availability across power modes

Detection time is programmable, and a digital filtering is available (tamper triggered after two false comparisons in four consecutive comparison samples).

Note: Timestamps can be generated when a tamper event occurs.

The internal tamper sources are listed in the table below.

Table 16. Internal tampers in TAMP

Tamper input	NOER bit number in TAMP_CR3	Tamper source
itamp1	0	V_{DD} domain voltage continuous monitoring, functional in Standby mode
itamp2	1	Temperature monitoring, functional in Standby mode
itamp3	2	LSE monitoring(1), functional in Standby mode
itamp4	3	HSE monitoring
itamp5	4	RTC calendar overflow (rtc_calovf)
itamp6	5	JTAG/SWD access
itamp7, 12, 13, 14	6, 11, 12, 15	Voltage monitoring (V_{CORE}, V_{REF+}), through ADC analog watchdog
itamp8	7	Monotonic counter overflow (generated internally)
itamp9	8	Fault generation for cryptographic peripherals (SAES, PKA, AES, RNG)
itamp11	10	IWDG timeout and potential tamper (IWDG reset when at least one enabled tamper flag is set)
itamp15	14	System fault detection

1. LSE missing or over frequency detection (> 2 MHz), Glitch filter (> 2 MHz).

Response to tampers

Each source of tamper in the device can be configured to trigger the following events:

- Generate an interrupt, capable of waking up the device from Stop and Standby modes (see TAMPxMSK bits in TAMP_CR2 register).
- Generate a hardware trigger for the low-power timers.
- Erase device secrets if the corresponding TAMPxPOM bit is cleared in TAMP_CR2 (for tamper pins) or TAMP_CR3 (for internal tamper). These erasable secrets are:
  - symmetric keys stored in backup registers (x32), in AES, HASH, and OTFDEC (encrypted flash memory regions are read as 0)
  - asymmetric keys stored in PKA SRAM, erased when V_{DD} is present
  - other secrets stored in SRAM2 and CPU instruction cache memory (SRAM2 erased when V_{DD} is present)
  - nonvolatile information used to derive the DHUK in SAES is 0-ed until complete SRAM2 erase

Read/write accesses by software to all these secrets can be blocked, by setting the BKBLOCK bit in TAMP_CR2. The device secrets access is possible only when BKBLOCK is cleared, and no tamper flag is set for any enabled tamper source.

If V_{DD} is not present, the secrets that are erased when V_{DD} is present are erased only at the next V_{DD} power on.

Note: Device secret erase is also triggered by setting the BKERASE bit in TAMP_CR2, or by performing a PRODUCT_STATE regression as defined in Section 3.8.1.

Device secrets are not reset by a system reset or when the device wakes up from Standby mode.

Software filtering mechanism

Each tamper source can be configured not to launch an immediate erase, by setting the corresponding TAMPxPOM bit in TAMP_CR2 (for external tamper pin) or TAMP_CR3 (for internal tamper).

In such situation, when the tamper flag is raised, access to below secrets is blocked until all tamper flags are cleared:

- DHUK in SAES: fixed to a dummy value
- Backup registers, SRAM2: read as 0, write-ignored
- AES, SAES, and HASH peripherals: automatically reset by RCC
- PKA peripheral: reset, with memory use blocked (meaning PKA not usable)

Once the application, notified by the tamper event, analyzes the situation, there are two possible cases:

- The application launches secrets erase with a software command (confirmed tamper).
- The application clears the flags to release secrets blocking (false tamper).

Note: If the tamper software fails to react to such a tamper flag, an IWDG reset triggers an automatic erasing of secrets.

Tamper detection and low-power modes

The effect of low-power modes on a tamper detection are summarized in the table below.

Table 17. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect on tamper detection features. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on tamper detection features, except for level detection with filtering and active tamper modes that remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Stop mode.
Standby	No effect on tamper detection features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Standby mode.

3.7 Secure storage

A critical feature of security systems is how long-term keys are stored, protected, and provisioned. Such keys are typically used for loading a boot image, or handling of critical user data.

- nonvolatile areas dedicated to secure storage, access controlled based on HDPL, TrustZone with one 8 kilo-bytes area (*Section 7.5: Option-byte key management*)
- Write-only key registers in the AES engines (*Section 34: AES hardware accelerator (AES)*)
- Secure storage on STM32G56x can rely on the SAES engine to encrypt/decrypt the stored data, to benefit of DHUK properties (*Section 34: AES hardware accelerator*

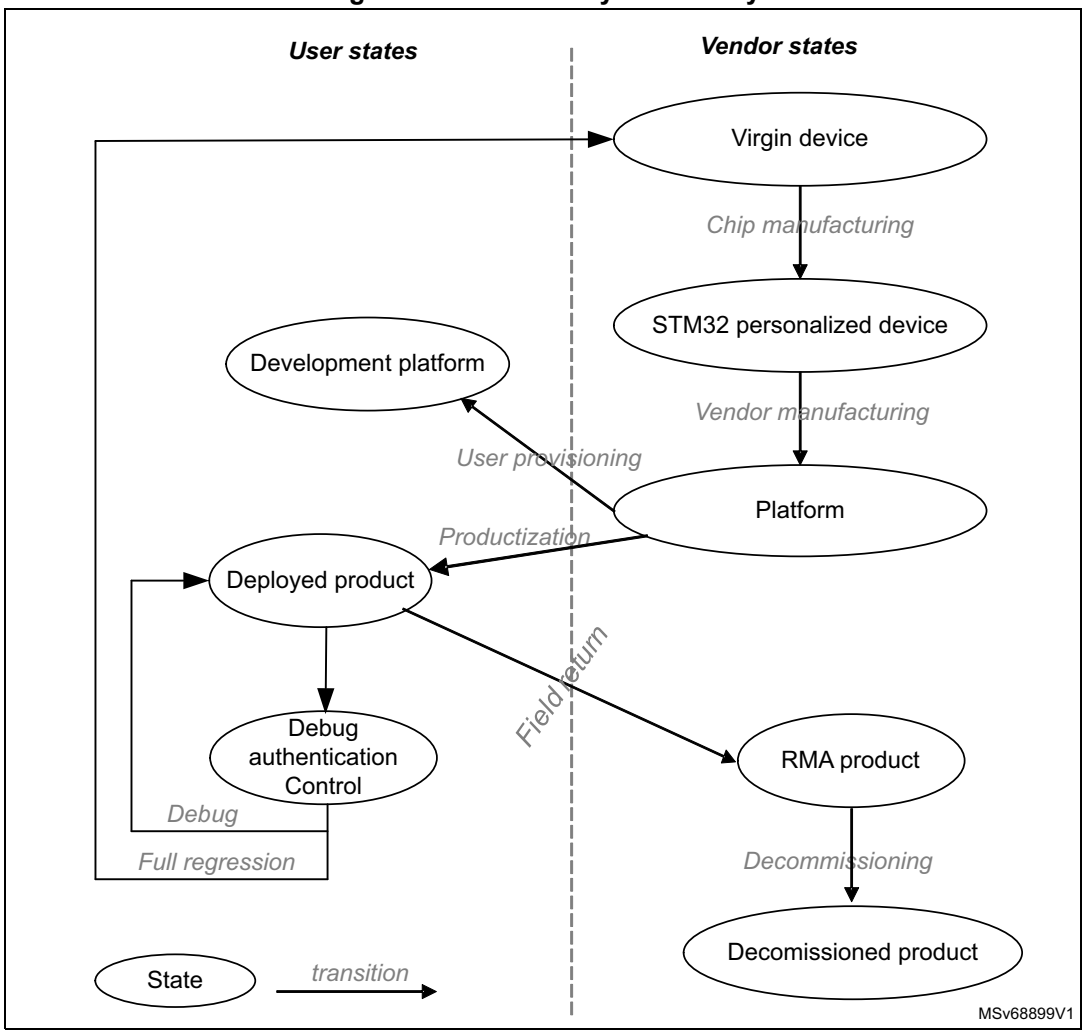
(AES), *Section 14: System configuration, boot, and security (SBS)*). DHUK offers several security properties to protect data:

- Temporal Isolation: data encrypted/decrypted relying on SAES+DHUK benefit of the variation of the DHUK for each different isolated area with HDPL0, HDPL1, HDPL2, HDPL3 (*Section 60: Debug support (DBG)*)
- Runtime Isolation: data encrypted/decrypted relying on SAES+DHUK benefit of the variation of DHUK for each compartment (CID) combined with TrustZone (HNONSEC). (*Section 5: Global TrustZone® controller (GTZC)*)
- DHUK (based on RHUK, which is also based on PUF (Physically Unclonable Function): hardware secret nonvolatile, unique per device keys (allowing anticloning)
- EPOCH counters: 3 counters to increment during regression (allows protection data against rollback)
- Coupling and chaining bridge (CCB) provide secure handling of stored asymmetric keys (*Section 32: Coupling and chaining bridge (CCB)*)

3.8 Product life cycle

A typical IoT device life cycle is summarized in *Figure 9*. For each step, the devices propose secure life cycle management mechanisms embedded in the hardware.

Figure 9. Device life-cycle security



Additional details on the various phases and associated transitions, found either at the vendor or end-user premises, are summarized in the table below.

Table 18. Main product life-cycle transitions

Transitions	Description
Device manufacturing	ST creates new STM32 devices, always checking for manufacturing defects. During this process, the device is provisioned with ROM firmware, and a public ID. STM32G56x provisioning also includes a secure firmware install (SFI) unique key pair, not implemented on STM32G55x/54x/53x.
Vendor manufacturing	One (or more) vendor is responsible for the platform assembly, initialization, and provisioning before delivery to the end user. This end user can use the final product (“productization” transition) or he/she can use the platform for software development (“user provisioning” transition).
Productization	The end user gets a product ready for use. All security functions of the platform are enabled, the debugging/testing features are restricted/disabled, and unique boot entry to immutable code is enforced.

Table 18. Main product life-cycle transitions (continued)

Transitions	Description
User provisioning	Platform vendor prepares an individual platform for development, not to be connected to a production cloud network.
Field return	The product is returned for analysis to field return centers. Analysis is possible by opening it partially, or by launching a partial or a full regression. Such accesses require provisioning, including certificates provided by the vendor. STMicroelectronics does not perform analysis on defective parts in closed or locked states.

The features described hereafter contribute to secure the device life cycle.

3.8.1 Life-cycle management

The product life cycle allows to control access to different assets (code and data) of the product, including during development, manufacturing, and after sales.

It allows to provision the product with different distribution models taking care on the code and data provisioned.

Table 19. Typical product life-cycle phases

PRODUCT_STATE + DebugState		Debug (default configuration)	Comments
Open	Device open	Secure(1) and nonsecure	Allows to develop the product, as it provides the debug of the code. Boot address must target a secure area when TrustZone is enabled. Using the boot pin allows to launch the bootloader. Using boot pin and BOOT_UBE allows to launch user flash code through ST-iROT.
Provisioning	Debug partially opened (only nonsecure)	HDPL3 + nonsecure only	Allows to manage the provisioning of the product (partial or full). It allows to launch secure firmware Install, or bootloader to provision the product. Boot address must target a secure area when TrustZone is enabled. Boot on SRAM is not permitted.
iROT-Provisioned	Debug partially opened (only nonsecure)	HDPL3 + nonsecure only	Assumes that immutable root of trust is installed, including its configuration (code, option bytes, secure storage). Boot address must target a secure area when TrustZone is enabled. Boot on SRAM is not permitted.
TZ-Closed	Debug partially opened (only nonsecure)	HDPL3 + nonsecure only	Assumes that the Secure OS is installed in TrustZone (this state exist only if TZEN = 0xB4). Debug is opened for nonsecure applications
Closed	Debug closed	No debug + debug authentication control	Assumes the product configuration as finalized. It allows to support debug authentication for in-the-field repair (read the dedicated application note).
Locked	Debug locked	None	Assumes the product configuration as finalized. The debug authentication is not permitted. The product is definitively in this state.

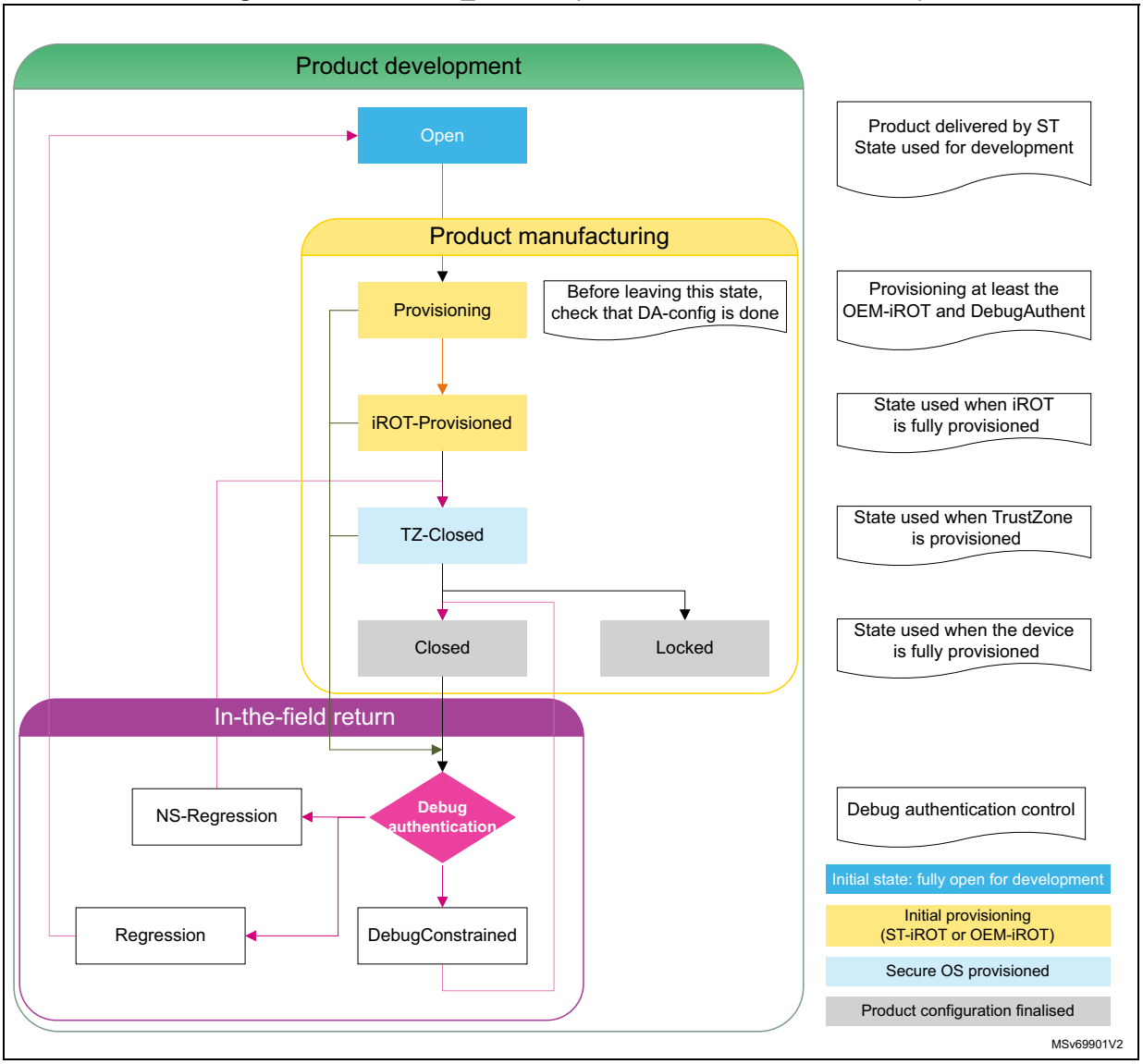
Table 19. Typical product life-cycle phases (continued)

PRODUCT_STATE + DebugState		Debug (default configuration)	Comments
NS-Regression	Debug closed	No debug	This is a temporary (but nonvolatile) state to manage the partial regression to the TZ-Closed, removing all nonsecure code and data including in secure storage.
Regression	Debug closed	No debug	This is a temporary state (but nonvolatile) to manage the full regression to Open state, removing all user flash code and data including in secure storage (HDPL1 to HDPL3).
Debug Constrained	Debug opened following debug authentication permissions	Depends upon permissions	This is a temporary state (until power-on-reset) to manage debug based on the permissions acquired by the debug authentication protocol.

1. Debug is not available when executing RSS code.

The supported transitions, summarized in the figure below, can be requested (when available) through the debug interface or via the system bootloader.

Figure 10. PRODUCT_STATEs (full TrustZone activated view)



The full view of the life cycle includes the debug authentication part, to help product maintenance to manage field returns (in-the-field centers). The debug authentication control is ensured thanks to a protocol based on Arm PSA ADAC specification. It allows controlling that the host has a trusted certificate with permissions. Permissions definition allows a lot of flexibility, in our default model we consider: Full or Partial regression, or to open the debug for nonsecure.

When TrustZone is disabled, the state TZ-Closed cannot be set. For details on the debug authentication feature, refer to the corresponding application note.

The protocol includes a “DISCOVER” command allowing to get basic informations of the target: manufacturer, product name, device identifier, protocol version supported and crypto scheme for authentication (this help selecting between certificates and password).

3.8.2 Recommended product settings

To ease the product maintenance (in-the-field), we recommend to take benefit of the feature called debug authentication control. This enables the maintenance of product activating the debug and makes possible to manage regressions while considering the security of the sensible information.

This implies the following actions:

- set the PRODUCT_STATE in Closed state when the product has been configured
- When debug authentication control is based on certificates, **provision the OEM-PublicKey and the SOC-Mask**. Setting the SOC-Mask to propose only the nonsecure-debug and partial and full regression guarantee to be compliant with the Arm PSA security model.
- When debug authentication control is based on password, only the HASH of the password must be provisioned.

A dedicated application note is available to help configuring the platform, the host tools and to generate certificates.

3.8.3 Software intellectual property protection

As described in *Section 7.6.11: Product state transitions,* the hardware PRODUCT_STATE mechanism controls the accesses to secrets provisioned in the device. The protection of these secrets are defined in the table below.

Table 20. Software intellectual property protection with PRODUCT_STAT

PRODUCT_STATE protection level		Secret protection
Open	Device open	No special protections.
Provisioning	Debug partially opened (only nonsecure)	All areas protected with HDPL1 cannot be dumped, debugged or traced. The iROT can setup a higher level of protection.
iROT-Provisioned	Debug partially opened (only nonsecure)	All areas protected with HDPL1 cannot be dumped, debugged or traced. The iROT can setup a higher level of protection.
TZ-Closed	Debug partially opened (only nonsecure)	All peripherals and memories mapped as secure during secure boot cannot be dumped, debugged or traced
Closed	No debug	No debug possible except through debug authentication.
Locked	No debug	All data and code stored in the device or encrypted in external flash memory cannot be dumped clear-text, debugged or traced.

3.8.4 Software intellectual property protection with OTFDEC

The feature described in this section is supported by STM32G56x, not implemented on STM32G55x/54x/53x.

As described in *Section 39: On-the-fly decryption engine (OTFDEC),* the OTFDEC associated with the XSPI1 is able to decrypt on the fly, the encrypted images stored in external SPI flash memories.

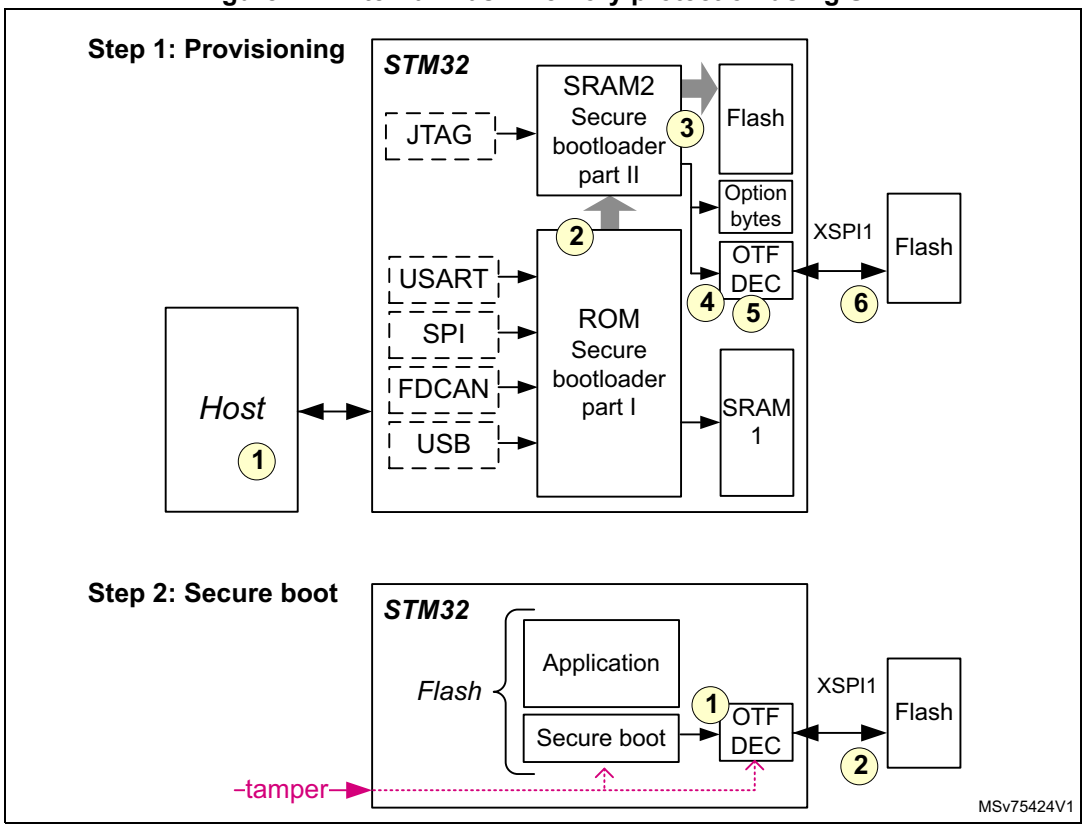
Thanks to this feature, the devices allow the installation of intellectual properties in one of the following ways:

- Over the air, with the image already encrypted with a key provisioned in the device
- Through a provisioning host located in a trusted or a non-trusted environment/facility

Figure 11 illustrates this last case, with the provisioning, in a non-trusted environment, of software intellectual properties both in the embedded flash memory and in an external SPI flash memory (encrypted).

Note: Since the OTFDEC uses the AES in counter mode (CTR) to achieve the lowest possible latency, each time the content of one encrypted region is changed, the corresponding cryptographic context (key or initialization vector) must be changed. This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR flash memory.

Figure 11. External Flash memory protection using SFI



Provisioning

Assuming the device is virgin, the first step is to provision both flash memories, as detailed below:

1. The user creates a SFI image, composed of:
   - encrypted internal firmware and data (including external flash memory drivers)
   - encrypted external firmware and data AES key (up to 4)

  - encrypted external firmware and data image

2. The secure bootloader stored in the system memory, loads the second part of the secure bootloader in SRAM2, through the supported communication ports (USART, SPI, FDCAN, USB, and JTAG). This second part runs in the secure SRAM2 and is responsible for executing the SFI process, applying the SFI protocol thanks to the commands received through the above mentioned supported communication ports.
3. The internal flash memory is programmed with decrypted option bytes, internal firmware and data, and external firmware and data AES keys. Alternatively, device unique external firmware AES keys can be used instead of such global keys.
4. The OTFDEC is properly initialized with encrypted region information, including the corresponding external firmware and data AES key.
5. Running the SFI process, chunks of encrypted external firmware and data image are decrypted in the device, then re-encrypted in the OTFDEC.
6. After a chunk OTFDEC re-encryption, the user external flash memory programmer is responsible for programming the last encrypted chunks to the external SPI flash memories through the XSPI1.

Secure boot

After provisioning, each time the device initializes on a trusted firmware, the following actions are required:

1. Secure-boot firmware executes, programming the external firmware and data AES keys to the OTFDEC write-only key registers, along with the other needed information.
2. The application reads or executes the encrypted external firmware and data through the XSPI1 in memory mapped mode, unless a tamper event is detected. In this case, all OTFDEC keys are erased and encrypted regions are read as 0 until the OTFDEC is properly initialized.

For more information on SFI solutions for the devices, refer to the AN4992 “*STM32 MCUs secure firmware install (SFI) overview”.*

3.8.5 Other software intellectual property protections

The device additional protections to software intellectual property are:

- Invasive attacks such as physical tampering or perturbation are countered by detection then decommissioning of the device before the detected attack succeeds (refer to *Section 50: Tamper and backup registers (TAMP)*).
- Noninvasive attacks, such as side channel attacks, are countered by not leaking secret information via side channels (such as timing, power, and EM radiations) on STM32G56x(refer to *Section 35: Secure AES coprocessor (SAES)* and *Section 32: Coupling and chaining bridge (CCB)*).

4 Boot modes

At startup, a BOOT0 pin and BOOTADD/SBOOTADD option bytes are used to select the boot memory address that includes:

- Boot from any address in user flash memory
- Boot from system memory
  - Bootloader
  - ST immutable root of trust (ST-iROT)
  - Root security service (RSS)
  - Debug authentication library (RSS-DA)

Note: RSS resources are supported by STM32G55x/56x, not implemented on STM32G53x/54x.

Note: ST-iROT resources are supported by STM32G56x, not implemented on STM32G53x/54x/55x.

Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. It is used to reprogram the flash memory by using USART, SPI, FDCAN, I2C, or USB through the DFU (device firmware upgrade) protocol.

Refer to the application note *STM32 microcontroller system memory boot mode (*AN2606).

Embedded root security services (RSS)

The embedded RSS are located in the secure information block, programmed by ST during production.

Refer to the application note *Overview secure firmware install (SFI) (*AN4992).

Embedded immutable root of trust (ST-iROT)

The embedded ST-iROT in the system memory, is programmed by ST during production. ST-iROT is the immutable root of trust managing the secure boot and secure install of the first updatable level to execute in a boot sequence.

Embedded debug authentication (ST-DA)

The embedded ST-DA in the system memory, is programmed by ST during production. ST-DA is the library that manages the debug authentication protocol by allowing to securely reopen the debug or to launch regressions on secured products in the field.

The tables below provide the detail of the boot mode products when the TrustZone is, respectively, disabled (TZEN = 0xC3) and enabled (TZEN = 0xB4).

Table 21. Boot mode when TrustZone is disabled (TZEN = 0xC3)

PRODUCT_STATE	BOOT0 pin	BOOT_UBE FLASH_OPTSR [29:22]	Boot address option-byte selection	Boot area	ST programmed default value
Open	0	NA	BOOTADD	Boot address defined by user option byte BOOTADD(1)	Flash: 0x0800 0000
	1	NA	NA	Bootloader	Bootloader
Provisioning	x	NA	NA	RSS(1)	RSS
Provisioned, Closed, Locked	x	NA	BOOTADD	Boot address defined by user option byte BOOTADD	Flash: 0x0800 0000

1. Jump to Bootloader can be enforced alternatively by using RSSCMD in SBS.

Table 22. Boot mode when TrustZone is enabled (TZEN = 0xB4) for STM32G53x/54x/55x

PRODUCT_STATE	BOOT0 pin	BOOT_UBE FLASH_OPTSR [29:22]	Boot address option-byte selection	Boot area	ST programmed default value
Open	0	NA	SBOOTADD	Boot address defined by user option byte SBOOTADD	Flash: 0x0C00 0000
	1	NA	NA	Bootloader	Bootloader
Provisioned, Provisioning, TZ_Closed, Closed, Locked	x	NA	SBOOTADD	Boot address defined by user option byte SBOOTADD	Flash: 0x0C00 0000

Table 23. Boot mode when TrustZone is enabled (TZEN = 0xB4) for STM32G56x

PRODUCT_STATE	BOOT0 pin	BOOT_UBE FLASH_OPTSR [29:22]	Boot address option-byte selection	Boot area	ST programmed default value
Open	0	x	SBOOTADD	Boot address defined by user option byte SBOOTADD	Flash: 0x0C00 0000
-	1	0xB4	NA	Bootloader	Bootloader
-	1	0xC3	NA	ST-iROT	ST-iROT
Provisioning	x	NA	NA	RSS	RSS

Table 23. Boot mode when TrustZone is enabled (TZEN = 0xB4) for STM32G56x (continued)

PRODUCT_STATE	BOOT0 pin	BOOT_UBE FLASH_OPTSR [29:22]	Boot address option-byte selection	Boot area	ST programmed default value
Provisioned, TZ_Closed, Closed, Locked	x	0xC3	ST-iROT	ST-iROT	ST-iROT
		0xB4	SBOOTADD	Boot address defined by user option byte SBOOTADD	Flash: 0x0C00 0000

When TrustZone is enabled (TZEN = 0xB4), the boot space must be in secure area. SBOOTADD option bytes are used to select the boot secure memory address. A unique boot entry option can be selected by setting the SBOOT_LOCK option bit.

5 Global TrustZone® controller (GTZC)

5.1 GTZC introduction

The global TrustZone controller (GTZC) block that contains the following subblocks:

- **TZSC**: TrustZone security controller

  Defines the secure/privileged state of slave peripherals, and controls the subregion area size and properties for the watermark memory peripheral controller (MPCWM). The TZSC informs some peripherals (such as RCC or GPIOs) about the secure status of each securable peripheral, by sharing with RCC and I/O logic.
- **MPCBB**: memory protection controller - block based

  Configures the internal RAM in a TrustZone-system product having segmented SRAM (pages of 512 bytes) with programmable-security and privileged attributes.
- **TZIC**: TrustZone illegal access controller

  Gathers all illegal access events in the system, and generates secure interrupts towards NVIC.

These subblocks are used to configure TrustZone system security in a product having bus agents with programmable-security and privileged attributes such as:

- On-chip RAM with programmable secure and/or privileged blocks (pages)
- AHB and APB peripherals with programmable security and/or privileged access
- Off-chip memories with secure and/or privileged areas

5.2 GTZC main features

- Three independent 32-bit AHB interfaces for TZSC, TZIC, and MPCBB
- TZIC accessible only with secure transactions
- Secure and nonsecure access supported for privileged and unprivileged parts of TZSC and MPCBB
- Set of registers to define product security settings:
  - Secure/privileged blocks for internal SRAMs
  - Secure/privileged regions for external memories
  - Secure/privileged access mode for securable peripherals
  - Secure/privileged access mode for securable masters

GTZC TrustZone system architecture

The Armv8-M supports security per TrustZone-M model with isolation between:

- A secure world, where usually security sensitive applications are run and critical resources are located
- A nonsecure or public world (such as the usual non secure operating system and user space)

The TrustZone architecture is extended beyond AHB and Armv8-M with:

- AHB/APB bridge used as secure gate to block or propagate secure/nonsecure and privileged/unprivileged transaction towards APB agents

- PPC (peripheral protection controller) used as secure gate to block or propagate secure/nonsecure and privileged/unprivileged transaction towards AHB agents
- TrustZone block-based MPC firewalls used as secure gate to filter secure/non secure, privileged/unprivileged access towards internal SRAMs
- TrustZone watermark MPC firewalls used as secure gate to filter secure/non secure, privileged/unprivileged access towards external memories

AHB and APB peripherals can be categorized as:

- **privileged**: peripherals protected by AHB/APB firewall stub that is controlled from TZSC to define privilege properties
- **secure**: peripherals always protected by an AHB/APB firewall stub. These peripherals are always secure (such as TZIC)
- **securable**: peripherals protected by an AHB/APB firewall stub that is controlled from TZSC to define security properties (optional)
- **nonsecure and unprivileged:** peripherals connected directly to AHB/APB interconnect without any secure gate
- **TrustZone-aware**: peripherals connected directly to the AHB or APB bus and implementing a specific TrustZone behavior (such as a subset of registers being secure). TrustZone-aware AHB masters always drive HNONSEC signal according to their security mode (such as Armv8-M core or DMA)

AHB securable masters can be configured in the TZSC to be secure/nonsecure and/or privileged/unprivileged.

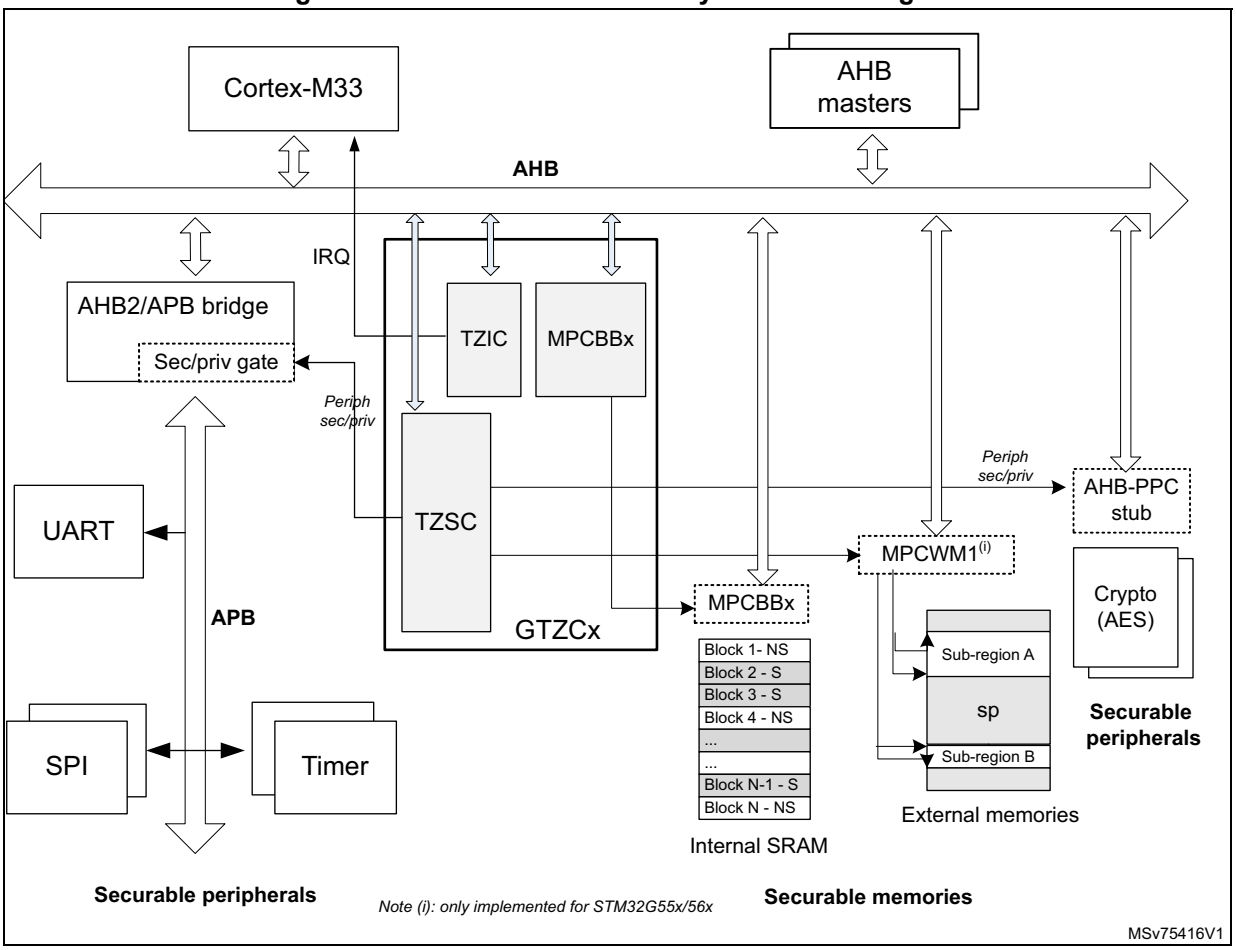
Application information

The TZSC, MPCBB, and TZIC can be used in one of the following ways:

- Programmed only during secure boot, locked and not changed afterwards
- Dynamically re-programmed when using specific application code or secure kernel (microvisor). When not locked, MPC secure blocks or region size can be changed by secure software executing from the secure FLASH region or secure SRAM. The same applies to the GTZC1_TZSC_SECCFGRx/PRIVCFGRx registers that define secure/privileged state of each peripheral.

The Armv8-M security architecture with secure, securable, and TrustZone-aware peripherals is shown in *Figure 12*.

Figure 12. GTZC in Armv8-M subsystem block diagram



5.3 GTZC implementation

The STM32G56x/55x/54x/53x devices embed one instance of GTZC.

Table 24. GTZC features

GTZC subblocks	GTZC1
GTZC1_TZSC	X
GTZC1_TZIC	X
GTZC1_MPCBBx subblock on STM32G53x/54x devices	MPCBB1/2
GTZC1_MPCBBx subblock on STM32G55x/56x devices	MPCBB1/2/3

The table below shows the address offset of GTZC subblocks versus GTZC base address (refer to *Section 2.3: Memory organization* for GTZC1 base address).

Table 25. GTZC1 subblock address offset

GTZC1 subblock	Address offset
GTZC1_TZSC	0x0
GTZC1_TZIC	0x400
GTZC1_MPCBB1	0x800
GTZC1_MPCBB2	0xC00
GTZC1_MPCBB3	0x1000

The table below describes the characteristics of the available MPCWM.

Table 26. MPCWM resource assignment

GTZC	MPC	Target memory interface	Number of sec/nonsec and priv/unpriv regions	Watermark granularity (bytes)	STM32G53x/54x	STM32G55x/56x
GTZC1	MPCWM1	XSPI1	2	128 K	-	X

The table below describes the characteristics of the available MPCBB.

Table 27. MPCBB resource assignment for STM32G53x/54x

GTZC	MPC	Resource	Memory size (Kbytes)	Block size (bytes)	Number of blocks	Number of super-blocks
GTZC1	MPCBB1	SRAM1	20	512	40	2
	MPCBB2	SRAM2	40		80	3

Table 28. MPCBB resource assignment for STM32G55x/56x

GTZC	MPC	Resource	Memory size (Kbytes)	Block size (bytes)	Number of blocks	Number of super-blocks
GTZC1	MPCBB1	SRAM1	32	512	64	4
	MPCBB2	SRAM2	64		128	8
	MPCBB3	SRAM3	16		32	2

5.4 GTZC functional description

5.4.1 GTZC block diagram

Figure 13 describes the combined feature of TZSC, MPCBB, and TZIC. Each subblock is controlled by its own AHB configuration port.

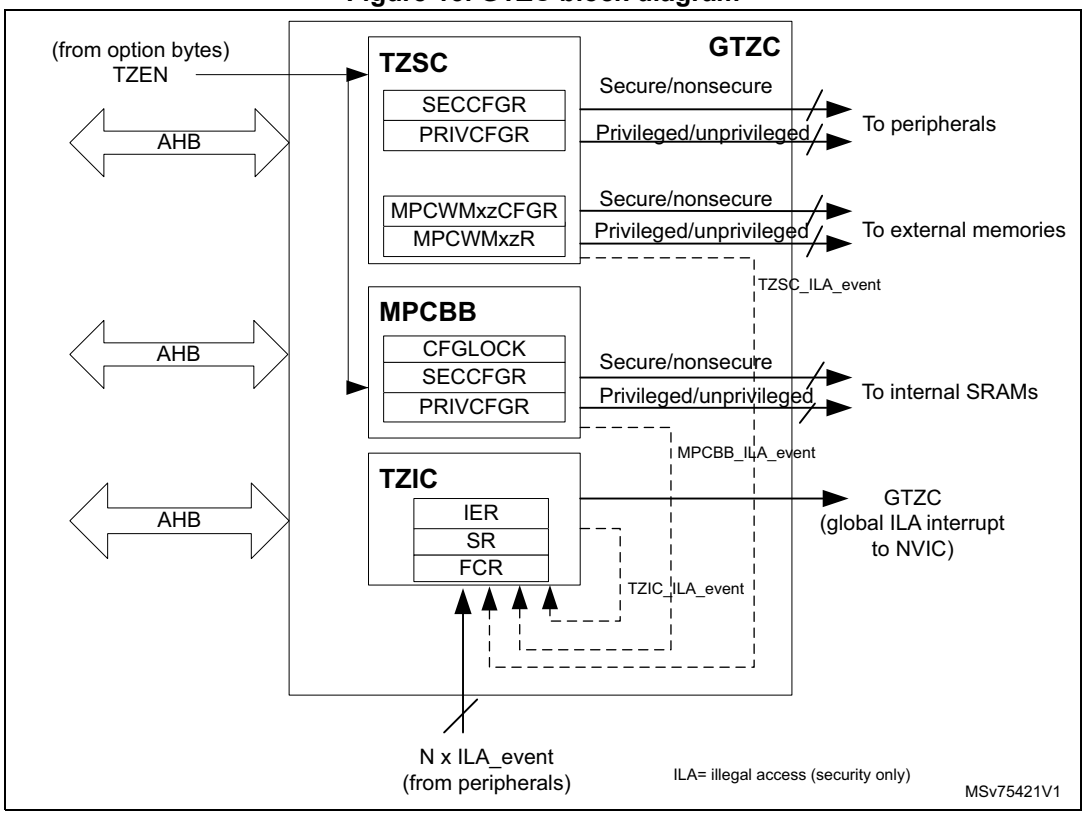
The TZSC defines which peripheral is secure and/or privileged. The privileged configuration bit of a peripheral can be modified by a secure privileged transaction when the peripheral is configured as secure. Otherwise, a privileged transaction (nonsecure) is sufficient.

On the opposite, the secure configuration bit of a peripheral can be modified only with a secure privileged transaction if the peripheral is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The secure configuration bit of a given RAM block can be modified only with a secure privileged transaction if the same block is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The TZIC gathers illegal events generated within the system when an illegal access is detected. TZIC can then generate a secure interrupt towards the CPU if needed.

Figure 13. GTZC block diagram



5.4.2 Illegal access definition

Three different types of illegal access exist:

- Illegal nonsecure access

  Any nonsecure transaction trying to write a secure resource is considered as illegal, hence the addressed resource generates an illegal access interrupt for illegal write access and a bus error for illegal fetch access. There are some exceptions on secure and privileged configuration registers: the latter ones authorize non secure read access to secure registers (see GTZC1_TZSC_SECCFGRx/GTZC1_TZSC_PRIVCFGRx).

- Illegal secure access

  Any secure transaction trying to access nonsecure block in internal block-based SRAM or watermarked memory is considered as illegal.

  A correct TZIC setting allows the capture of the associated event and then generates the GTZC_IRQn interrupt to the NVIC. This applies for read, write, and execute access.

  Concerning the MPCBB controller, there is an option to ignore secure data read/write access on nonsecure SRAM blocks, by setting the SRWILADIS bit in the GTZC1_MPCBBz_CR register. Secure read and write data transactions are then allowed on nonsecure SRAM blocks, while secure execution access remains not allowed.

  Any secure execute transaction trying to access a nonsecure peripheral register is considered as illegal and generates a bus error.

- Illegal unprivileged access

  Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal read and write accesses. The addressed resource follows a silent-fail behavior, returning all 0 data for read and ignoring any write. No bus error is generated. A bus error is generated when an unprivileged execute transaction tries to access a privileged memory.

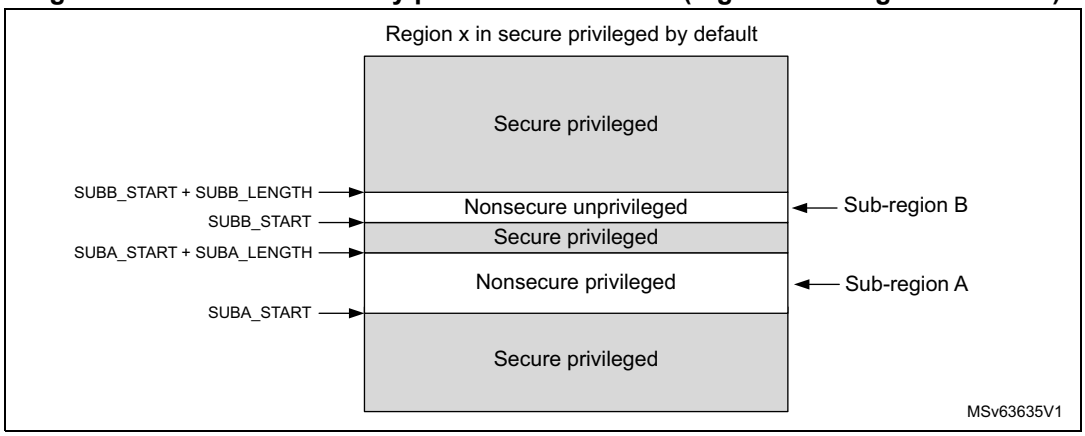
5.4.3 TrustZone security controller (TZSC)

The TZSC is composed of a configurable set of registers, providing the following features:

- Control of secure and privileged state for all peripherals, done through:
  - GTZC1_TZSC_SECCFGRx registers to control AHB/APB firewall stubs for the securable peripherals
  - GTZC1_TZSC_PRIVCFGRx registers to control AHB/APB firewall stubs for the privileged peripherals
- For STM32G55x/56x watermark XSPI memory protection controller, two independent regions can be defined and the following fields are used to program:
  - the start of the first protected subregion on the external memory: SUBA_START[10:0]
  - the length of the first protected subregion on the external memory: SUBA_LENGTH[11:0]
  - the start of the second protected subregion on the external memory: SUBB_START[10:0]
  - the length of the second protected subregion on the external memory: SUBB_LENGTH[11:0]

A control register for each subregion can be used to enable/disable the watermark memory protection controller, and to define the right attributes of each subregion.

Figure 14. Watermark memory protection controller (region x/subregions A and B)



In *Figure 14*, region x represents the external memory. Secure and privileged attributes of subregions A and B are independently configurable. When no subregions are defined or enabled on the region x, the default attribute of the region x is set as secure-privileged.

The following tables describe the secure/privileged properties of the common area of subregions A and B, when an overlap exists.

Table 29. Secure properties of subregions A and B

Subregion A	Subregion B	Properties of overlapped region A and B
nonsecure	nonsecure	nonsecure
nonsecure	Secure	nonsecure
Secure	nonsecure	nonsecure
Secure	Secure	Secure

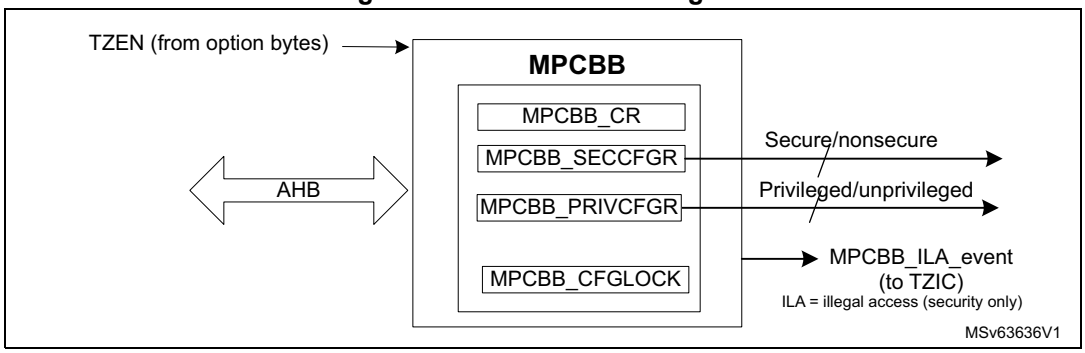
Table 30. Privileged properties of subregions A and B

Subregion A	Subregion B	Properties of overlapped region A and B
Unprivileged	Unprivileged	Unprivileged
Unprivileged	Privileged	Unprivileged
Privileged	Unprivileged	Unprivileged
Privileged	Privileged	Privileged

5.4.4 Memory protection controller - block based (MPCBB)

The MPCBB is composed of a configurable set of registers allowing to define security and privileged policy for internal SRAMs. The security and privileged policy can be individually configured per each 512-byte block.

Figure 15. MPCBB block diagram



To set up the MPCBB, the following actions are needed (for example at boot time):

- Secure firmware must define which memory blocks are secure by setting the correct bits in GTZC1_MPCBBz_SECCFGRx.
- Privileged firmware must define which memory blocks are privileged by setting the correct bits in GTZC1_MPCBBz_PRIVCFGRx.

An MPCBB super-block is made of 32 consecutive blocks. For each super-block, secure application can lock all related security/privileged bits using the correct bits in GTZC1_MPCBBz_CFGLOCK. This lock remains active until the next system reset.

*Note: The block size is 512 bytes. The super-block size is 512 * 32 = 16 Kbytes.*

5.4.5 TrustZone illegal access controller (TZIC)

The TZIC concentrates all illegal access source events. It is used only when the system is TrustZone enabled (TZEN = 0xB4).

TZIC allows to trace (flag) the event that triggered the secure illegal access interrupt. Register masks (GTZC1_TZIC_IERx) are available to filter unwanted event. On unmasked illegal event, TZIC generates the GTZC_IRQn interrupt to the NVIC.

For each illegal event source, a status flag and a clear bit exist (respectively within GTZC1_TZIC_SRx and GTZC1_TZIC_FCRx). The reset value of mask registers (GTZC1_TZIC_IERx) is such that all events are masked.

5.4.6 Power-on/reset state

The power-on and reset state of the TZSC clear to 0 all bits of GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx, meaning that all securable peripherals are respectively set to nonsecure and unprivileged.

For internal SRAMx (x = 1 to 3), all GTZC1_MPCBBz_SECCFGRx and GTZC1_MPCBBz_PRIVCFGRx are set:

- to 0xFFFF FFFF, making these internal memories block secure and privileged by default when TrustZone security is enabled at system level (TZEN = 0xB4).
- to 0x0000 0000, making these internal memories block nonsecure and unprivileged by default when TrustZone security is disabled at system level (TZEN = 0xC3)

For STM32G55x/56x external memory, GTZC1_TZSC_MPCWM1zR registers are set:

- to 0x0000 0000, making these memories secure and privileged by default when TrustZone security is enabled at system level (TZEN = 0xB4).
- to 0x0800 0000, making these memories nonsecure and non-privileged by default when TrustZone security is disabled at system level (TZEN = 0xC3)

Secure boot code can then program the security settings, making components secure or not as needed.

5.5 GTZC interrupts

TZIC is a secure peripheral, thus it systematically generates an illegal access event when accessed by a nonsecure access. The MPCBB and TZSC are TrustZone-aware peripherals, meaning that secure and nonsecure registers coexist within the peripheral.

Table 31. GTZC interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
GTZC	Illegal access	All flags in GTZC1_TZIC_SRx	All bits in GTZC1_TZIC_IERx	Write 1 in the bit GTZC1_TZIC_FCRx	Yes	Yes	No

5.6 GTZC1 TZSC registers

All registers are accessed only by words (32-bit).

5.6.1 GTZC1 TZSC control register (GTZC1_TZSC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LCK**: lock the configuration of GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx until next reset

This bit is cleared by default, and once set, it cannot be reset until system reset.

0: configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx not locked

1: configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx locked

5.6.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1)

Address offset: 0x010

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR signal is set to 1. If a given PRIV bit is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DTSSEC	Res.	TIM11SEC	TIM10SEC	TIM9SEC	DAC2SEC	DAC1SEC	Res.	Res.	UART6SEC	CRSSEC	I3C1SEC	I2C2SEC	I2C1SEC	UART5SEC
	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4SEC	USART3SEC	USART2SEC	SPI3SEC	SPI2SEC	IWDGSEC	WWDGSEC	TIM14SEC	TIM13SEC	TIM12SEC	TIM7SEC	TIM6SEC	TIM5SEC	TIM4SEC	TIM3SEC	TIM2SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DTSSEC**: secure access mode for DTS

0: nonsecure

1: secure

Bit 29 Reserved, must be kept at reset value.

Bit 28 **TIM11SEC**: secure access mode for TIM11

0: nonsecure

1: secure

Bit 27 **TIM10SEC**: secure access mode for TIM10

0: nonsecure

1: secure

Bit 26 **TIM9SEC**: secure access mode for TIM9

0: nonsecure

1: secure

Bit 25 **DAC2SEC**: secure access mode for DAC2

0: nonsecure

1: secure

Bit 24 **DAC1SEC**: secure access mode for DAC1
0: nonsecure
1: secure

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **UART6SEC**: secure access mode for UART6
0: nonsecure
1: secure

Bit 20 **CRSSEC**: secure access mode for CRS
0: nonsecure
1: secure
Note: depending on device

Bit 19 **I3C1SEC**: secure access mode for I3C1
0: nonsecure
1: secure

Bit 18 **I2C2SEC**: secure access mode for I2C2
0: nonsecure
1: secure

Bit 17 **I2C1SEC**: secure access mode for I2C1
0: nonsecure
1: secure

Bit 16 **UART5SEC**: secure access mode for UART5
0: nonsecure
1: secure

Bit 15 **UART4SEC**: secure access mode for UART4
0: nonsecure
1: secure

Bit 14 **USART3SEC**: secure access mode for USART3
0: nonsecure
1: securew
Note: depending on device

Bit 13 **USART2SEC**: secure access mode for USART2
0: nonsecure
1: secure

Bit 12 **SPI3SEC**: secure access mode for SPI3
0: nonsecure
1: secure
Note: depending on device

Bit 11 **SPI2SEC**: secure access mode for SPI2
0: nonsecure
1: secure

Bit 10 **IWDGSEC**: secure access mode for IWDG
0: nonsecure
1: secure

Bit 9 **WWDGSEC**: secure access mode for WWDG
0: nonsecure
1: secure

Bit 8 **TIM14SEC**: secure access mode for TIM14
0: nonsecure
1: secure

Bit 7 **TIM13SEC**: secure access mode for TIM13
0: nonsecure
1: secure

Bit 6 **TIM12SEC**: secure access mode for TIM12
0: nonsecure
1: secure

Bit 5 **TIM7SEC**: secure access mode for TIM7
0: nonsecure
1: secure

Bit 4 **TIM6SEC**: secure access mode for TIM6
0: nonsecure
1: secure

Bit 3 **TIM5SEC**: secure access mode for TIM5
0: nonsecure
1: secure
Note: depending on device

Bit 2 **TIM4SEC**: secure access mode for TIM5
0: nonsecure
1: secure
Note: depending on device

Bit 1 **TIM3SEC**: secure access mode for TIM3
0: nonsecure
1: secure
Note: depending on device

Bit 0 **TIM2SEC**: secure access mode for TIM2
0: nonsecure
1: secure

5.6.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2)

Address offset: 0x014

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR signal is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP3SEC	COMP2SEC	COMP1SEC	LPTIM1SEC	I2C4SEC	I2C3SEC	LPUART1SEC	TIM26SEC	OPAMP3SEC	OPAMP2SEC	OPAMP1SEC	TIM20SEC	USBSEC	HRTIM1SEC	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	TIM8SEC	SPI1SEC	TIM1SEC	Res.	Res.	Res.	Res.	UCPD1SEC	FDCAN3SEC	FDCAN2SEC	FDCAN1SEC
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **COMP3SEC**: secure access mode for COMP3
0: nonsecure
1: secure

Bit 30 **COMP2SEC**: secure access mode for COMP2
0: nonsecure
1: secure

Bit 29 **COMP1SEC**: secure access mode for COMP1
0: nonsecure
1: secure

Bit 28 **LPTIM1SEC**: secure access mode for LPTIM1
0: nonsecure
1: secure

Bit 27 **I2C4SEC**: secure access mode for I2C4
0: nonsecure
1: secure
Note: depending on device

Bit 26 **I2C3SEC**: secure access mode for I2C3
0: nonsecure
1: secure
Note: depending on device

Bit 25 **LPUART1SEC**: secure access mode for LPUART1
0: nonsecure
1: secure

Bit 24 **TIM26SEC**: secure access mode for TIM26
0: nonsecure
1: secure

Bit 23 **OPAMP3SEC**: secure access mode for OPAMP3
0: nonsecure
1: secure

Bit 22 **OPAMP2SEC**: secure access mode for OPAMP2
0: nonsecure
1: secure

Bit 21 **OPAMP1SEC**: secure access mode for OPAMP1
0: nonsecure
1: secure

Bit 20 **TIM20SEC**: secure access mode for TIM20
0: nonsecure
1: secure
Note: depending on device

Bit 19 **USBSEC**: secure access mode for USB
0: nonsecure
1: secure
Note: depending on device

Bit 18 **HRTIM1SEC**: secure access mode for HRTIM1
0: nonsecure
1: secure

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **TIM17SEC**: secure access mode for TIM17
0: nonsecure
1: secure

Bit 13 **TIM16SEC**: secure access mode for TIM16
0: nonsecure
1: secure

Bit 12 **TIM15SEC**: secure access mode for TIM15
0: nonsecure
1: secure

Bit 11 **USART1SEC**: secure access mode for USART1
0: nonsecure
1: secure

Bit 10 **TIM8SEC**: secure access mode for TIM8
0: nonsecure
1: secure

Bit 9 **SPI1SEC**: secure access mode for SPI1
0: nonsecure
1: secure

Bit 8 **TIM1SEC**: secure access mode for TIM1
0: nonsecure
1: secure

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **UCPD1SEC**: secure access mode for UCPD
0: nonsecure
1: secure
Note: depending on device

Bit 2 **FDCAN3SEC**: secure access mode for FDCAN3
0: nonsecure
1: secure

Caution: The FDCAN message RAM is secure only when all FDCAN instances are configured as secure.

Note: depending on device

Bit 1 **FDCAN2SEC**: secure access mode for FDCAN2
0: nonsecure
1: secure

Caution: The FDCAN message RAM is secure only when all FDCAN instances are configured as secure.

Note: depending on device

Bit 0 **FDCAN1SEC**: secure access mode for FDCAN1
0: nonsecure
1: secure

Caution: The FDCAN message RAM is secure only when all FDCAN instances are configured as secure.

5.6.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3)

Address offset: 0x018

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCBSEC	DAC10SEC	DAC9SEC	DAC8SEC	RAMCFGSEC	Res.	XSPI1SEC	DAC7SEC	DAC6SEC	DAC5SEC	PKASEC	SAESSEC	RNGSEC	HASHSEC	AESSEC	ADC3SEC
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC12SEC	DCACHESEC	ICACHESEC	FMAC2SEC	FMAC1SEC	CORDIC1SEC	CRCSEC	POSENC2SEC	POSENC1SEC	TIM27SEC	VREFBUFSEC	COMP8SEC	COMP7SEC	COMP6SEC	COMP5SEC	COMP4SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CCBSEC**: secure access mode for CCB
0: nonsecure
1: secure

Bit 30 **DAC10SEC**: secure access mode for DAC10
0: nonsecure
1: secure

Bit 29 **DAC9SEC**: secure access mode for DAC9
0: nonsecure
1: secure

Bit 28 **DAC8SEC**: secure access mode for DAC8
0: nonsecure
1: secure

Bit 27 **RAMCFGSEC**: secure access mode for RAMCFG
0: nonsecure
1: secure

Bit 26 Reserved, must be kept at reset value.

Bit 25 **XSPI1SEC**: secure access mode for XSPI1
0: nonsecure
1: secure
Note: depending on device

Bit 24 **DAC7SEC**: secure access mode for DAC7
0: nonsecure
1: secure

Bit 23 **DAC6SEC**: secure access mode for DAC6
0: nonsecure
1: secure

Bit 22 **DAC5SEC**: secure access mode for DAC5
0: nonsecure
1: secure

Bit 21 **PKASEC**: secure access mode for PKA
0: nonsecure
1: secure

Bit 20 **SAESSEC**: secure access mode for SAES
0: nonsecure
1: secure
Note: depending on device

Bit 19 **RNGSEC**: secure access mode for RNG
0: nonsecure
1: secure

Bit 18 **HASHSEC**: secure access mode for HASH
0: nonsecure
1: secure

Bit 17 **AESSEC**: secure access mode for AES
0: nonsecure
1: secure

Bit 16 **ADC3SEC**: secure access mode for ADC3
0: nonsecure
1: secure
Note: depending on device

Bit 15 **ADC12SEC**: secure access mode for ADC1 and ADC2
0: nonsecure
1: secure

Bit 14 **DCACHESEC**: secure access mode for DCACHE
0: nonsecure
1: secure
Note: depending on device

Bit 13 **ICACHESEC**: secure access mode for ICACHE
0: nonsecure
1: secure

Bit 12 **FMAC2SEC**: secure access mode for register of FMAC2
0: nonsecure
1: secure
Note: depending on device

Bit 11 **FMAC1SEC**: secure access mode for register of FMAC1
0: nonsecure
1: secure

Bit 10 **CORDIC1SEC**: secure access mode for CORDIC1
0: nonsecure
1: secure

Bit 9 **CRCSEC**: secure access mode for CRC
0: nonsecure
1: secure

Bit 8 **POSENC2SEC**: secure access mode for POSENC2
0: nonsecure
1: secure
Note: depending on device

Bit 7 **POSENC1SEC**: secure access mode for POSENC1
0: nonsecure
1: secure
Note: depending on device

Bit 6 **TIM27SEC**: secure access mode for TIM27
0: nonsecure
1: secure

Bit 5 **VREFBUFSEC**: secure access mode for VREFBUF
0: nonsecure
1: secure

Bit 4 **COMP8SEC**: secure access mode for COMP8
0: nonsecure
1: secure

Bit 3 **COMP7SEC**: secure access mode for COMP7
0: nonsecure
1: secure

Bit 2 **COMP6SEC**: secure access mode for COMP6
0: nonsecure
1: secure

Bit 1 **COMP5SEC**: secure access mode for COMP5
0: nonsecure
1: secure

Bit 0 **COMP4SEC**: secure access mode for COMP4
0: nonsecure
1: secure

5.6.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1)

Address offset: 0x020

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR signal is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DTSPRIV	Res.	TIM11PRIV	TIM10PRIV	TIM9PRIV	DAC2PRIV	DAC1PRIV	Res.	Res.	UART6PRIV	CRSPRIV	I3C1PRIV	I2C2PRIV	I2C1PRIV	UART5PRIV
	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4PRIV	USART3PRIV	USART2PRIV	SPI3PRIV	SPI2PRIV	IWDGPRIV	WWDGPRIV	TIM14PRIV	TIM13PRIV	TIM12PRIV	TIM7PRIV	TIM6PRIV	TIM5PRIV	TIM4PRIV	TIM3PRIV	TIM2PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DTSPRIV**: privileged access mode for DTS
0: unprivileged
1: privileged

Bit 29 Reserved, must be kept at reset value.

Bit 28 **TIM11PRIV**: privileged access mode for TIM11
0: unprivileged
1: privileged

Bit 27 **TIM10PRIV**: privileged access mode for TIM10
0: unprivileged
1: privileged

Bit 26 **TIM9PRIV**: privileged access mode for TIM9
0: unprivileged
1: privileged

Bit 25 **DAC2PRIV**: privileged access mode for DAC2
0: unprivileged
1: privileged

Bit 24 **DAC1PRIV**: privileged access mode for DAC1
0: unprivileged
1: privileged

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **UART6PRIV**: privileged access mode for UART6
0: unprivileged
1: privileged

Bit 20 **CRSPRIV**: privileged access mode for CRS
0: unprivileged
1: privileged

Bit 19 **I3C1PRIV**: privileged access mode for I3C1
0: unprivileged
1: privileged
Note: depending on device

Bit 18 **I2C2PRIV**: privileged access mode for I2C2
0: unprivileged
1: privileged

Bit 17 **I2C1PRIV**: privileged access mode for I2C1
0: unprivileged
1: privileged

Bit 16 **UART5PRIV**: privileged access mode for UART5
0: unprivileged
1: privileged

Bit 15 **UART4PRIV**: privileged access mode for UART4
0: unprivileged
1: privileged

Bit 14 **USART3PRIV**: privileged access mode for USART3
0: unprivileged
1: privileged
Note: depending on device

Bit 13 **USART2PRIV**: privileged access mode for USART2
0: unprivileged
1: privileged

Bit 12 **SPI3PRIV**: privileged access mode for SPI3
0: unprivileged
1: privileged
Note: depending on device

Bit 11 **SPI2PRIV**: privileged access mode for SPI2
0: unprivileged
1: privileged

Bit 10 **IWDGPRIV**: privileged access mode for IWDG
0: unprivileged
1: privileged

Bit 9 **WWDGPRIV**: privileged access mode for WWDG
0: unprivileged
1: privileged

Bit 8 **TIM14PRIV**: privileged access mode for TIM14
0: unprivileged
1: privileged

Bit 7 **TIM13PRIV**: privileged access mode for TIM13
0: unprivileged
1: privileged

Bit 6 **TIM12PRIV**: privileged access mode for TIM12
0: unprivileged
1: privileged

Bit 5 **TIM7PRIV**: privileged access mode for TIM7
0: unprivileged
1: privileged

Bit 4 **TIM6PRIV**: privileged access mode for TIM6
0: unprivileged
1: privileged

Bit 3 **TIM5PRIV**: privileged access mode for TIM5
0: unprivileged
1: privileged
Note: depending on device

Bit 2 **TIM4PRIV**: privileged access mode for TIM4
0: unprivileged
1: privileged
Note: depending on device

Bit 1 **TIM3PRIV**: privileged access mode for TIM3
0: unprivileged
1: privileged
Note: depending on device

Bit 0 **TIM2PRIV**: privileged access mode for TIM2
0: unprivileged
1: privileged

5.6.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR signal is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP3PRIV	COMP2PRIV	COMP1PRIV	LPTIM1PRIV	I2C4PRIV	I2C3PRIV	LPUART1PRIV	TIM26PRIV	OPAMP3PRIV	OPAMP2PRIV	OPAMP1PRIV	TIM20PRIV	USBPRIV	HRTIM1PRIV	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV	TIM8PRIV	SPI1PRIV	TIM1PRIV	Res.	Res.	Res.	Res.	UCPDPRIV	FDCAN3PRIV	FDCAN2PRIV	FDCAN1PRIV
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **COMP3PRIV**: privileged access mode for COMP3
0: unprivileged
1: privileged

Bit 30 **COMP2PRIV**: privileged access mode for COMP2
0: unprivileged
1: privileged

Bit 29 **COMP1PRIV**: privileged access mode for COMP1
0: unprivileged
1: privileged

Bit 28 **LPTIM1PRIV**: privileged access mode for LPTIM1
0: unprivileged
1: privileged

Bit 27 **I2C4PRIV**: privileged access mode for I2C4
0: unprivileged
1: privileged
Note: depending on device

Bit 26 **I2C3PRIV**: privileged access mode for I2C3
0: unprivileged
1: privileged
Note: depending on device

Bit 25 **LPUART1PRIV**: privileged access mode for LPUART1
0: unprivileged
1: privileged

Bit 24 **TIM26PRIV**: privileged access mode for TIM26
0: unprivileged
1: privileged

Bit 23 **OPAMP3PRIV**: privileged access mode for OPAMP3
0: unprivileged
1: privileged

Bit 22 **OPAMP2PRIV**: privileged access mode for OPAMP2
0: unprivileged
1: privileged

Bit 21 **OPAMP1PRIV**: privileged access mode for OPAMP1
0: unprivileged
1: privileged

Bit 20 **TIM20PRIV**: privileged access mode for TIM20
0: unprivileged
1: privileged
Note: depending on device

Bit 19 **USBPRIV**: privileged access mode for USB
0: unprivileged
1: privileged
Note: depending on device

Bit 18 **HRTIM1PRIV**: privileged access mode for HRTIM1
0: unprivileged
1: privileged

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **TIM17PRIV**: privileged access mode for TIM17
0: unprivileged
1: privileged

Bit 13 **TIM16PRIV**: privileged access mode for TIM16
0: unprivileged
1: privileged

Bit 12 **TIM15PRIV**: privileged access mode for TIM15
0: unprivileged
1: privileged

Bit 11 **USART1PRIV**: privileged access mode for USART1
0: unprivileged
1: privileged

Bit 10 **TIM8PRIV**: privileged access mode for TIM8
0: unprivileged
1: privileged

Bit 9 **SPI1PRIV**: privileged access mode for SPI1
0: unprivileged
1: privileged

Bit 8 **TIM1PRIV**: privileged access mode for TIM1
0: unprivileged
1: privileged

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **UCPDPRIV**: privileged access mode for UCPD
0: unprivileged
1: privileged
Note: depending on device

Bit 2 **FDCAN3PRIV**: privileged access mode for FDCAN3
0: unprivileged
1: privileged

Caution: The FDCAN message RAM is privileged only when all FDCAN instances are configured as privileged

Note: depending on device

Bit 1 **FDCAN2PRIV**: privileged access mode for FDCAN2
0: unprivileged
1: privileged

Caution: The FDCAN message RAM is privileged only when all FDCAN instances are configured as privileged

Note: depending on device

Bit 0 **FDCAN1PRIV**: privileged access mode for FDCAN1
0: unprivileged
1: privileged

Caution: The FDCAN message RAM is privileged only when all FDCAN instances are configured as privileged

5.6.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3)

Address offset: 0x028

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR signal is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCBPRIV	DAC10PRIV	DAC9PRIV	DAC8PRIV	RAMCFGPRIV	Res.	XSPI1PRIV	DAC7PRIV	DAC6PRIV	DAC5PRIV	PKAPRIV	SAESPRIV	RNGPRIV	HASHPRIV	AESPRIV	ADC3PRIV
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC12PRIV	DCACHEPRIV	ICACHEPRIV	FMAC2PRIV	FMAC1PRIV	CORDIC1PRIV	CRCPRIV	POSENC2PRIV	POSENC1PRIV	TIM27PRIV	VREFBUFPRIV	COMP8PRIV	COMP7PRIV	COMP6PRIV	COMP5PRIV	COMP4PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CCBPRIV**: privileged access mode for CCB
0: unprivileged
1: privileged

Bit 30 **DAC10PRIV**: privileged access mode for DAC10
0: unprivileged
1: privileged

Bit 29 **DAC9PRIV**: privileged access mode for DAC9
0: unprivileged
1: privileged

Bit 28 **DAC8PRIV**: privileged access mode for DAC8
0: unprivileged
1: privileged

Bit 27 **RAMCFGPRIV**: privileged access mode for RAMCFG
0: unprivileged
1: privileged

Bit 26 Reserved, must be kept at reset value.

Bit 25 **XSPI1PRIV**: privileged access mode for XSPI1
0: unprivileged
1: privileged
Note: depending on device

Bit 24 **DAC7PRIV**: privileged access mode for DAC7
0: unprivileged
1: privileged

Bit 23 **DAC6PRIV**: privileged access mode for DAC6
0: unprivileged
1: privileged

Bit 22 **DAC5PRIV**: privileged access mode for DAC5
0: unprivileged
1: privileged

Bit 21 **PKAPRIV**: privileged access mode for PKA
0: unprivileged
1: privileged

Bit 20 **SAESPRIV**: privileged access mode for SAES
0: unprivileged
1: privileged
Note: depending on device

Bit 19 **RNGPRIV**: privileged access mode for RNG
0: unprivileged
1: privileged

Bit 18 **HASHPRIV**: privileged access mode for HASH
0: unprivileged
1: privileged

Bit 17 **AESPRIV**: privileged access mode for AES
0: unprivileged
1: privileged
Note: depending on device

Bit 16 **ADC3PRIV**: privileged access mode for ADC3
0: unprivileged
1: privileged
Note: depending on device

Bit 15 **ADC12PRIV**: privileged access mode for ADC1 and ADC2
0: unprivileged
1: privileged

Bit 14 **DCACHEPRIV**: privileged access mode for DCACHE
0: unprivileged
1: privileged
Note: depending on device

Bit 13 **ICACHEPRIV**: privileged access mode for ICACHE
0: unprivileged
1: privileged

Bit 12 **FMAC2PRIV**: privileged access mode for register of FMAC2
0: unprivileged
1: privileged
Note: depending on device

Bit 11 **FMAC1PRIV**: privileged access mode for register of FMAC1
0: unprivileged
1: privileged

Bit 10 **CORDIC1PRIV**: privileged access mode for CORDIC1
0: unprivileged
1: privileged

Bit 9 **CRCPRIV**: privileged access mode for CRC
0: unprivileged
1: privileged

Bit 8 **POSENC2PRIV**: privileged access mode for POSENC2
0: unprivileged
1: privileged
Note: depending on device

Bit 7 **POSENC1PRIV**: privileged access mode for POSENC1
0: unprivileged
1: privileged
Note: depending on device

Bit 6 **TIM27PRIV**: privileged access mode for TIM27
0: unprivileged
1: privileged

Bit 5 **VREFBUFPRIV**: privileged access mode for VREFBUF
0: unprivileged
1: privileged

Bit 4 **COMP8PRIV**: privileged access mode for COMP8
0: unprivileged
1: privileged

Bit 3 **COMP7PRIV**: privileged access mode for COMP7
0: unprivileged
1: privileged

Bit 2 **COMP6PRIV**: privileged access mode for COMP6
0: unprivileged
1: privileged

Bit 1 **COMP5PRIV**: privileged access mode for COMP5
0: unprivileged
1: privileged

Bit 0 **COMP4PRIV**: privileged access mode for COMP4
0: unprivileged
1: privileged

5.6.8 GTZC1 TZSC memory x subregion z watermark configuration register (GTZC1_TZSC_MPCWM1zCFGR) (z = A to B)

Address offset: Block A: 0x40

Address offset: Block B: 0x48

Reset value: 0x0000 0000

Secure privilege access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
						rw	rw							rs	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **PRIV**: Privileged subregion z of base region
This bit is taken into account only if SREN is set.
0: Privileged and unprivileged accesses are granted in subregion z.
1: Only privileged accesses are granted in subregion z.

Bit 8 **SEC**: Secure subregion z of base region
This bit is taken into account only if SREN is set.
0: Only nonsecure data accesses are granted to subregion z.
1: Only secure data accesses are granted to subregion z.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **SRLOCK**: subregion z lock
This bit, once set, can be cleared only by a system reset.
0: GTZC1_TZSC_MPCWM1CFGR, GTZC1_TZSC_MPCWM1AR and GTZC1_TZSC_MPCWM1BR can be written.
1: Writes to GTZC1_TZSC_MPCWM1CFGR, GTZC1_TZSC_MPCWM1AR and GTZC1_TZSC_MPCWM1BR are ignored.

Bit 0 **SREN**: subregion z enable

0: subregion z is disabled. Access control of base region applies to any access between this subregion start- and end-addresses.

1: subregion z is enabled. Access control defined in GTZC1_TZSC_MPCWM1_CFGR applies to any access between this subregion start- and end-addresses, defined in GTZC1_TZSC_MPCWM1AR and GTZC1_TZSC_MPCWM1BR.

Note: Watermark controlled external memories start fully nonsecure/unprivileged at reset when TZEN = 0xC3. When TZEN = 0xB4, they start fully secure/fully privileged (inverted reset-value).

5.6.9 GTZC1 TZSC subregion A watermark register (GTZC1_TZSC_MPCWM1AR)

Address offset: 0x44

Reset value: 0x0000 0000

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0800 0000 when TZEN = 0xC3.

Secure privilege access only.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBA_LENGTH is applied automatically.

When an overlap of subregion A and B exists, secure/privileged attributes of both subregions apply on the common section (see *Section 5.4.3*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SUBA_LENGTH[11:0]**: Length of subregion A in base region

This field defines the length of the subregion A, to be multiplied by the granularity defined in *Table 26*.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBA_LENGTH is applied automatically. If SUBA_LENGTH = 0, subregion A is disabled (SREN bit in GTZC1_TZSC_MPCMW1ACFGR is cleared).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **SUBA_START[10:0]**: Start of subregion A in base region

This field defines the address offset of the subregion A, to be multiplied by the granularity defined in *Table 26*, versus the start of the region.

Watermark controlled external memories start fully nonsecure at reset when TZEN = 0xC3. When TZEN = 0xB4, they start fully secure (inverted reset value).

5.6.10 GTZC1 TZSC subregion B watermark register (GTZC1_TZSC_MPCWM1BR)

Address offset: 0x4C

Reset value: 0x0000 0000

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0800 0000 when TZEN = 0xC3.

Secure privilege access only.

When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.

When an overlap of subregion A and B exists, secure/privileged attributes of both subregions apply on the common section (see *Section 5.4.3*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SUBB_START[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SUBB_LENGTH[11:0]**: Length of subregion B in base region

This field defines the length of the subregion B, to be multiplied by the granularity defined in *Table 26*.

When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.

If SUBB_LENGTH = 0, the subregion B is disabled (SREN bit in GTZC1_TZSC_MPCMW1BCFGR is cleared).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **SUBB_START[10:0]**: Start of subregion B in base region

This field defines the address offset of the subregion B, to be multiplied by the granularity defined in *Table 26*, versus the start of the region x.

Watermark controlled external memories start fully nonsecure at reset when TZEN = 0xC3. When TZEN = 0xB4, they start fully secure (inverted reset value).

5.6.11 GTZC1 TZSC register map

Table 32. GTZC1 TZSC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_TZSC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
	Reset value																																0
0x004-0x00C	Reserved	Reserved																															

Table 32. GTZC1 TZSC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	GTZC1_TZSC_ SECCFGR1	Res.	DTSSEC	Res.	TIM11SEC	TIM10SEC	TIM9SEC	DAC2SEC	DAC1SEC	Res.	Res.	UART6SEC	CRSSEC	I3C1SEC	I2C2SEC	I2C1SEC	UART5SEC	UART4SEC	USART3SEC	USART2SEC	SPI3SEC	SPI2SEC	IWDGSEC	WWDGSEC	TIM14SEC	TIM13SEC	TIM12SEC	TIM7SEC	TIM6SEC	TIM5SEC	TIM4SEC	TIM3SEC	TIM2SEC
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	GTZC1_TZSC_ SECCFGR2	COMP3SEC	COMP2SEC	COMP1SEC	LPTIM1SEC	I2C4SEC	I2C3SEC	LPUART1SEC	TIM26SEC	OPAMP3SEC	OPAMP2SEC	OPAMP1SEC	TIM20SEC	USBSEC	HRTIM1SEC	Res.	Res.	Res.	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	TIM8SEC	SPI1SEC	TIM1SEC	Res.	Res.	Res.	Res.	UCPD1SEC	FDCAN3SEC	FDCAN2SEC	FDCAN1SEC
	Reset value	0	0	0	0	0	0	0	0					0	0		0	0	0	0	0	0	0	0	0						0	0	0
0x018	GTZC1_TZSC_ SECCFGR3	CCBSEC	DAC10SEC	DAC9SEC	DAC8SEC	RAMCFGSEC	Res.	XSPI1SEC	DAC7SEC	DAC6SEC	DAC5SEC	PKASEC	SAESSEC	RNGSEC	HASHSEC	AESSEC	ADC3SEC	ADC12SEC	DCACHESEC	ICACHESEC	FMAC2SEC	FMAC1SEC	CORDIC1SEC	CRCSEC	POSENC2SEC	POSENC1SEC	TIM27SEC	VREFBUFSEC	COMP8SEC	COMP7SEC	COMP6SEC	COMP5SEC	COMP4SEC
	Reset value						0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x01C	Reserved	Reserved																															
0x020	GTZC1_TZSC_ PRIVCFGR1	Res.	DTSPRIV	Res.	TIM11PRIV	TIM10PRIV	TIM9PRIV	DAC2PRIV	DAC1PRIV	Res.	Res.	UART6PRIV	CRSPRIV	I3C1PRIV	I2C2PRIV	I2C1PRIV	UART5PRIV	UART4PRIV	USART3PRIV	USART2PRIV	SPI3PRIV	SPI2PRIV	IWDGPRIV	WWDGPRIV	TIM14PRIV	TIM13PRIV	TIM12PRIV	TIM7PRIV	TIM6PRIV	TIM5PRIV	TIM4PRIV	TIM3PRIV	TIM2PRIV
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	GTZC1_TZSC_ PRIVCFGR2	COMP3PRIV	COMP2PRIV	COMP1PRIV	LPTIM1PRIV	I2C4PRIV	I2C3PRIV	LPUART1PRIV	TIM26PRIV	OPAMP3PRIV	OPAMP2PRIV	OPAMP1PRIV	TIM20PRIV	USBPRIV	HRTIM1PRIV	Res.	Res.	Res.	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV	TIM8PRIV	SPI1PRIV	TIM1PRIV	Res.	Res.	Res.	Res.	UCPDPRIV	FDCAN3PRIV	FDCAN2PRIV	FDCAN1PRIV
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x028	GTZC1_TZSC_ PRIVFGR3	CCBPRIV	DAC10PRIV	DAC9PRIV	DAC8PRIV	RAMCFGPRIV	Res.	XSPI1PRIV	DAC7PRIV	DAC6PRIV	DAC5PRIV	PKAPRIV	SAESPRIV	RNGPRIV	HASHPRIV	AESPRIV	ADC3PRIV	ADC12PRIV	DCACHEPRIV	ICACHEPRIV	FMAC2PRIV	FMAC1PRIV	CORDIC1PRIV	CRCPRIV	POSENC2PRIV	POSENC1PRIV	TIM27PRIV	VREFBUFPRIV	COMP8PRIV	COMP7PRIV	COMP6PRIV	COMP5PRIV	COMP4PRIV
	Reset value						0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x02C-0x03C	Reserved	Reserved																															
0x040	GTZC1_TZSC_ MPCWM1ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x44	GTZC1_TZSC_ MPCWM1AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x048	GTZC1_TZSC_ MPCWM1BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x04C	GTZC1_TZSC_ MPCWM1BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 25: GTZC1 subblock address offset*.

5.7 GTZC1 TZIC registers

All registers are accessed only by words (32-bit).

5.7.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DTSIE	Res.	TIM11IE	TIM10IE	TIM9IE	DAC2IE	DAC1IE	Res.	Res.	UART6IE	CRSIE	I3C1IE	I2C2IE	I2C1IE	UART5IE
	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4IE	USART3IE	USART2IE	SPI3IE	SPI2IE	IWDGIE	WWDGIE	TIM14IE	TIM13IE	TIM12IE	TIM7IE	TIM6IE	TIM5IE	TIM4IE	TIM3IE	TIM2IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DTSIE**: illegal access interrupt enable for DTS
0: interrupt disabled
1: interrupt enabled

Bit 29 Reserved, must be kept at reset value.

Bit 28 **TIM11IE**: illegal access interrupt enable for TIM11
0: interrupt disabled
1: interrupt enabled

Bit 27 **TIM10IE**: illegal access interrupt enable for TIM10
0: interrupt disabled
1: interrupt enabled

Bit 26 **TIM9IE**: illegal access interrupt enable for TIM9
0: interrupt disabled
1: interrupt enabled

Bit 25 **DAC2IE**: illegal access interrupt enable for DAC2
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAC1IE**: illegal access interrupt enable for DAC1
0: interrupt disabled
1: interrupt enabled

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **UART6IE**: illegal access interrupt enable for UART6
0: interrupt disabled
1: interrupt enabled

Bit 20 **CRSIE**: illegal access interrupt enable for CRS
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 19 **I3C1IE**: illegal access interrupt enable for I3C1
0: interrupt disabled
1: interrupt enabled

Bit 18 **I2C2IE**: illegal access interrupt enable for I2C2
0: interrupt disabled
1: interrupt enabled

Bit 17 **I2C1IE**: illegal access interrupt enable for I2C1
0: interrupt disabled
1: interrupt enabled

Bit 16 **UART5IE**: illegal access interrupt enable for UART5
0: interrupt disabled
1: interrupt enabled

Bit 15 **UART4IE**: illegal access interrupt enable for UART4
0: interrupt disabled
1: interrupt enabled

Bit 14 **USART3IE**: illegal access interrupt enable for USART3
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 13 **USART2IE**: illegal access interrupt enable for USART2
0: interrupt disabled
1: interrupt enabled

Bit 12 **SPI3IE**: illegal access interrupt enable for SPI3
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 11 **SPI2IE**: illegal access interrupt enable for SPI2
0: interrupt disabled
1: interrupt enabled

Bit 10 **IWDGIE**: illegal access interrupt enable for IWDG
0: interrupt disabled
1: interrupt enabled

Bit 9 **WWDGIE**: illegal access interrupt enable for WWDG
0: interrupt disabled
1: interrupt enabled

Bit 8 **TIM14IE**: illegal access interrupt enable for TIM14
0: interrupt disabled
1: interrupt enabled

Bit 7 **TIM13IE**: illegal access interrupt enable for TIM13
0: interrupt disabled
1: interrupt enabled

Bit 6 **TIM12IE**: illegal access interrupt enable for TIM12
0: interrupt disabled
1: interrupt enabled

Bit 5 **TIM7IE**: illegal access interrupt enable for TIM7
0: interrupt disabled
1: interrupt enabled

Bit 4 **TIM6IE**: illegal access interrupt enable for TIM6
0: interrupt disabled
1: interrupt enabled

Bit 3 **TIM5IE**: illegal access interrupt enable for TIM5
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 2 **TIM4IE**: illegal access interrupt enable for TIM4
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 1 **TIM3IE**: illegal access interrupt enable for TIM3
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 0 **TIM2IE**: illegal access interrupt enable for TIM2
0: interrupt disabled
1: interrupt enabled

5.7.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2)

Address offset: 0x004

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP3IE	COMP2IE	COMP1IE	LPTIM1IE	I2C4IE	I2C3IE	LPUART1IE	TIM26IE	OPAMP3IE	OPAMP2IE	OPAMP1IE	TIM20IE	USBIE	HRTIM1IE	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TIM17IE	TIM16IE	TIM15IE	USART1IE	TIM8IE	SPI1IE	TIM1IE	Res.	Res.	Res.	Res.	UCPD1IE	FDCAN3IE	FDCAN2IE	FDCAN1IE
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **COMP3IE**: illegal access interrupt enable for COMP3
0: interrupt disabled
1: interrupt enabled

Bit 30 **COMP2IE**: illegal access interrupt enable for COMP2
0: interrupt disabled
1: interrupt enabled

Bit 29 **COMP1IE**: illegal access interrupt enable for COMP1
0: interrupt disabled
1: interrupt enabled

Bit 28 **LPTIM1IE**: illegal access interrupt enable for LPTIM1
0: interrupt disabled
1: interrupt enabled

Bit 27 **I2C4IE**: illegal access interrupt enable for I2C4
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 26 **I2C3IE**: illegal access interrupt enable for I2C3
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 25 **LPUART1IE**: illegal access interrupt enable for LPUART1
0: interrupt disabled
1: interrupt enabled

Bit 24 **TIM26IE**: illegal access interrupt enable for TIM26
0: interrupt disabled
1: interrupt enabled

Bit 23 **OPAMP3IE**: illegal access interrupt enable for OPAMP3
0: interrupt disabled
1: interrupt enabled

Bit 22 **OPAMP2IE**: illegal access interrupt enable for OPAMP2
0: interrupt disabled
1: interrupt enabled

Bit 21 **OPAMP1IE**: illegal access interrupt enable for OPAMP1
0: interrupt disabled
1: interrupt enabled

Bit 20 **TIM20IE**: illegal access interrupt enable for TIM20
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 19 **USBIE**: illegal access interrupt enable for USB
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 18 **HRTIM1IE**: illegal access interrupt enable for HRTIM1
0: interrupt disabled
1: interrupt enabled

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **TIM17IE**: illegal access interrupt enable for TIM17
0: interrupt disabled
1: interrupt enabled

Bit 13 **TIM16IE**: illegal access interrupt enable for TIM16
0: interrupt disabled
1: interrupt enabled

Bit 12 **TIM15IE**: illegal access interrupt enable for TIM15
0: interrupt disabled
1: interrupt enabled

Bit 11 **USART1IE**: illegal access interrupt enable for USART1
0: interrupt disabled
1: interrupt enabled

Bit 10 **TIM8IE**: illegal access interrupt enable for TIM8
0: interrupt disabled
1: interrupt enabled

Bit 9 **SPI1IE**: illegal access interrupt enable for SPI1
0: interrupt disabled
1: interrupt enabled

Bit 8 **TIM1IE**: illegal access interrupt enable for TIM1
0: interrupt disabled
1: interrupt enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **UCPD1IE**: illegal access interrupt enable for UCPD1
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 2 **FDCAN3IE**: illegal access interrupt enable for FDCAN3
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 1 **FDCAN2IE**: illegal access interrupt enable for FDCAN2
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 0 **FDCAN1IE**: illegal access interrupt enable for FDCAN1 and FDCAN message RAM
0: interrupt disabled
1: interrupt enabled

5.7.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3)

Address offset: 0x008

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCBIE	DAC10IE	DAC9IE	DAC8IE	RAMCFGIE	Res.	XSPI1IE	DAC7IE	DAC6IE	DAC5IE	PKAIE	SAESIE	RNGIE	HASHIE	AESIE	ADC3IE
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC12IE	DCACHEIE	ICACHEIE	FMAC2IE	FMAC1IE	CORDIC1IE	CRCIE	POSENC2IE	POSENC1IE	TIM27IE	VREFBUFIE	COMP8IE	COMP7IE	COMP6IE	COMP5IE	COMP4IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CCBIE**: illegal access interrupt enable for CCB
0: interrupt disabled
1: interrupt enabled

Bit 30 **DAC10IE**: illegal access interrupt enable for DAC10
0: interrupt disabled
1: interrupt enabled

Bit 29 **DAC9IE**: illegal access interrupt enable for DAC9
0: interrupt disabled
1: interrupt enabled

Bit 28 **DAC8IE**: illegal access interrupt enable for DAC8
0: interrupt disabled
1: interrupt enabled

Bit 27 **RAMCFGIE**: illegal access interrupt enable for RAMCFG
0: interrupt disabled
1: interrupt enabled

Bit 26 Reserved, must be kept at reset value.

Bit 25 **XSPI1IE**: illegal access interrupt enable for XSPI1
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 24 **DAC7IE**: illegal access interrupt enable for DAC7
0: interrupt disabled
1: interrupt enabled

Bit 23 **DAC6IE**: illegal access interrupt enable for DAC6
0: interrupt disabled
1: interrupt enabled

Bit 22 **DAC5IE**: illegal access interrupt enable for DAC5
0: interrupt disabled
1: interrupt enabled

Bit 21 **PKAIE**: illegal access interrupt enable for PKA
0: interrupt disabled
1: interrupt enabled

Bit 20 **SAESIE**: illegal access interrupt enable for SAES
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 19 **RNGIE**: illegal access interrupt enable for RNG
0: interrupt disabled
1: interrupt enabled

Bit 18 **HASHIE**: illegal access interrupt enable for HASH
0: interrupt disabled
1: interrupt enabled

Bit 17 **AESIE**: illegal access interrupt enable for AES
0: interrupt disabled
1: interrupt enabled

Bit 16 **ADC3IE**: illegal access interrupt enable for ADC3
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 15 **ADC12IE**: illegal access interrupt enable for ADC1 and ADC2
0: interrupt disabled
1: interrupt enabled

Bit 14 **DCACHEIE**: illegal access interrupt enable for DCACHE
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 13 **ICACHEIE**: illegal access interrupt enable for ICACHE
0: interrupt disabled
1: interrupt enabled

Bit 12 **FMAC2IE**: illegal access interrupt enable for register of FMAC2
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 11 **FMAC1IE**: illegal access interrupt enable for register of FMAC1
0: interrupt disabled
1: interrupt enabled

Bit 10 **CORDIC1IE**: illegal access interrupt enable for CORDIC1
0: interrupt disabled
1: interrupt enabled

Bit 9 **CRCIE**: illegal access interrupt enable for CRC
0: interrupt disabled
1: interrupt enabledh

Bit 8 **POSENC2IE**: illegal access interrupt enable for POSENC2
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 7 **POSENC1IE**: illegal access interrupt enable for POSENC1
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 6 **TIM27IE**: illegal access interrupt enable for TIM27
0: interrupt disabled
1: interrupt enabled

Bit 5 **VREFBUFIE**: illegal access interrupt enable for VREFBUF
0: interrupt disabled
1: interrupt enabled

Bit 4 **COMP8IE**: illegal access interrupt enable for COMP8
0: interrupt disabled
1: interrupt enabled

Bit 3 **COMP7IE**: illegal access interrupt enable for COMP7
0: interrupt disabled
1: interrupt enabled

Bit 2 **COMP6IE**: illegal access interrupt enable for COMP6
0: interrupt disabled
1: interrupt enabled

Bit 1 **COMP5IE**: illegal access interrupt enable for COMP5
0: interrupt disabled[2]
1: interrupt enabled

Bit 0 **COMP4IE**: illegal access interrupt enable for COMP4
0: interrupt disabled
1: interrupt enabled

5.7.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	MPCBB3_REGIE	SRAM3IE	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	Res.	Res.	Res.	Res.	Res.	XSPI1_MEMIE	TZIC1IE	TZSC1IE
		rw	rw	rw	rw	rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PLAY1IE	EXTIIE	RCCIE	PWRIE	TAMPIE	RTCIE	SBSIE	Res.	OTFDEC1IE	FLASH_REGIE	FLASHIE	GPDMA2IE	GPDMA1IE
			rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **MPCBB3_REGIE**: illegal access interrupt enable for MPCBB3 registers
0: interrupt disabled
1: interrupt enabled
Note: depending on device.

Bit 28 **SRAM3IE**: illegal access interrupt enable for SRAM3
0: interrupt disabled
1: interrupt enabled
Note: depending on device.

Bit 27 **MPCBB2_REGIE**: illegal access interrupt enable for MPCBB2 registers
0: interrupt disabled
1: interrupt enabled

Bit 26 **SRAM2IE**: illegal access interrupt enable for SRAM2
0: interrupt disabled
1: interrupt enabled

Bit 25 **MPCBB1_REGIE**: illegal access interrupt enable for MPCBB1 registers
0: interrupt disabled
1: interrupt enabled

Bit 24 **SRAM1IE**: illegal access interrupt enable for SRAM1
0: interrupt disabled
1: interrupt enabled

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **XSPI1_MEMIE**: illegal access interrupt enable for MPCWM1 (XSPI1) memory bank
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 17 **TZIC1IE**: illegal access interrupt enable for GTZC1 TZIC registers
0: interrupt disabled
1: interrupt enabled

Bit 16 **TZSC1IE**: illegal access interrupt enable for GTZC1 TZSC registers
0: interrupt disabled
1: interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLAY1IE**: illegal access interrupt enable for PLAY
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 11 **EXTIIE**: illegal access interrupt enable for EXTI
0: interrupt disabled
1: interrupt enabled

Bit 10 **RCCIE**: illegal access interrupt enable for RCC
0: interrupt disabled
1: interrupt enabled

Bit 9 **PWRIE**: illegal access interrupt enable for PWR
0: interrupt disabled
1: interrupt enabled

Bit 8 **TAMPIE**: illegal access interrupt enable for TAMP
0: interrupt disabled
1: interrupt enabled

Bit 7 **RTCIE**: illegal access interrupt enable for RTC
0: interrupt disabled
1: interrupt enabled

Bit 6 **SBSIE**: illegal access interrupt enable for SBS
0: interrupt disabled
1: interrupt enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC1IE**: illegal access interrupt enable for OTFDEC1
0: interrupt disabled
1: interrupt enabled
Note: depending on device

Bit 3 **FLASH_REGIE**: illegal access interrupt enable for FLASH memory
0: interrupt disabled
1: interrupt enabled

Bit 2 **FLASHIE**: illegal access interrupt enable for FLASH registers
0: interrupt disabled
1: interrupt enabled

Bit 1 **GPDMA2IE**: illegal access interrupt enable for GPDMA2
0: interrupt disabled
1: interrupt enabled

Bit 0 **GPDMA1IE**: illegal access interrupt enable for GPDMA1
0: interrupt disabled
1: interrupt enabled

5.7.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DTSF	Res.	TIM11F	TIM10F	TIM9F	DAC2F	DAC1F	Res.	Res.	UART6F	CRSF	I3C1F	I2C2F	I2C1F	UART5F
	r		r	r	r	r	r			r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4F	USART3F	USART2F	SPI3F	SPI2F	IWDGF	WWDGF	TIM14F	TIM13F	TIM12F	TIM7F	TIM6F	TIM5F	TIM4F	TIM3F	TIM2F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DTSF:** illegal access flag for DTS
0: no illegal access event
1: illegal access event

Bit 29 Reserved, must be kept at reset value.

Bit 28 **TIM11F:** illegal access flag for TIM11
0: no illegal access event
1: illegal access event

Bit 27 **TIM10F:** illegal access flag for TIM10
0: no illegal access event
1: illegal access event

Bit 26 **TIM9F:** illegal access flag for TIM9
0: no illegal access event
1: illegal access event

Bit 25 **DAC2F:** illegal access flag for DAC2
0: no illegal access event
1: illegal access event

Bit 24 **DAC1F:** illegal access flag for DAC1
0: no illegal access event
1: illegal access event

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **UART6F:** illegal access flag for UART6
0: no illegal access event
1: illegal access event

Bit 20 **CRSF:** illegal access flag for CRS
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 19 **I3C1F:** illegal access flag for I3C1
0: no illegal access event
1: illegal access event

Bit 18 **I2C2F:** illegal access flag for I2C2
0: no illegal access event
1: illegal access event

Bit 17 **I2C1F:** illegal access flag for I2C1
0: no illegal access event
1: illegal access event

Bit 16 **UART5F:** illegal access flag for UART5
0: no illegal access event
1: illegal access event

Bit 15 **UART4F:** illegal access flag for UART4
0: no illegal access event
1: illegal access event

Bit 14 **USART3F:** illegal access flag for USART3
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 13 **USART2F:** illegal access flag for USART2
0: no illegal access event
1: illegal access event

Bit 12 **SPI3F:** illegal access flag for SPI3
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 11 **SPI2F:** illegal access flag for SPI2
0: no illegal access event
1: illegal access event

Bit 10 **IWDGF:** illegal access flag for IWDG
0: no illegal access event
1: illegal access event

Bit 9 **WWDGF:** illegal access flag for WWDG
0: no illegal access event
1: illegal access event

Bit 8 **TIM14F:** illegal access flag for TIM14
0: no illegal access event
1: illegal access event

Bit 7 **TIM13F:** illegal access flag for TIM13
0: no illegal access event
1: illegal access event

Bit 6 **TIM12F:** illegal access flag for TIM12
0: no illegal access event
1: illegal access event

Bit 5 **TIM7F:** illegal access flag for TIM7
0: no illegal access event
1: illegal access event

Bit 4 **TIM6F:** illegal access flag for TIM6
0: no illegal access event
1: illegal access event

Bit 3 **TIM5F:** illegal access flag for TIM5
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 2 **TIM4F:** illegal access flag for TIM4
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 1 **TIM3F:** illegal access flag for TIM3
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 0 **TIM2F**: illegal access flag for TIM2
0: no illegal access event
1: illegal access event

5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2)

Address offset: 0x014

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP 3F	COMP 2F	COMP 1F	LPTIM1 F	I2C4F	I2C3F	LPUAR T1F	TIM26F	OPAMP 3F	OPAMP 2F	OPAMP 1F	TIM20F	USBF	HRTIM 1F	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TIM17F	TIM16F	TIM15F	USART 1F	TIM8F	SPI1F	TIM1F	Res.	Res.	Res.	Res.	UCPD1 F	FDCAN 3F	FDCAN 2F	FDCAN 1F
	r	r	r	r	r	r	r					r	r	r	r

Bit 31 **COMP3F**: illegal access flag for COMP3
0: no illegal access event
1: illegal access event

Bit 30 **COMP2F**: illegal access flag for COMP2
0: no illegal access event
1: illegal access event

Bit 29 **COMP1F**: illegal access flag for COMP1
0: no illegal access event
1: illegal access event

Bit 28 **LPTIM1F**: illegal access flag for LPTIM1
0: no illegal access event
1: illegal access event

Bit 27 **I2C4F**: illegal access flag for I2C4
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 26 **I2C3F**: illegal access flag for I2C3
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 25 **LPUART1F**: illegal access flag for LPUART1
0: no illegal access event
1: illegal access event

Bit 24 **TIM26F**: illegal access flag for TIM26
0: no illegal access event
1: illegal access event

Bit 23 **OPAMP3F**: illegal access flag for OPAMP3
0: no illegal access event
1: illegal access event

Bit 22 **OPAMP2F**: illegal access flag for OPAMP2
0: no illegal access event
1: illegal access event

Bit 21 **OPAMP1F**: illegal access flag for OPAMP1
0: no illegal access event
1: illegal access event

Bit 20 **TIM20F**: illegal access flag for TIM20
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 19 **USBF**: illegal access flag for USB
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 18 **HRTIM1F**: illegal access flag for HRTIM1
0: no illegal access event
1: illegal access event

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **TIM17F**: illegal access flag for TIM17
0: no illegal access event
1: illegal access event

Bit 13 **TIM16F**: illegal access flag for TIM16
0: no illegal access event
1: illegal access event

Bit 12 **TIM15F**: illegal access flag for TIM15
0: no illegal access event
1: illegal access event

Bit 11 **USART1F**: illegal access flag for USART1
0: no illegal access event
1: illegal access event

Bit 10 **TIM8F**: illegal access flag for TIM8
0: no illegal access event
1: illegal access event

Bit 9 **SPI1F**: illegal access flag for SPI1
0: no illegal access event
1: illegal access event

Bit 8 **TIM1F**: illegal access flag for TIM1
0: no illegal access event
1: illegal access event

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **UCPD1F**: illegal access flag for UCPD
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 2 **FDCAN3F**: illegal access flag for FDCAN3
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 1 **FDCAN2F**: illegal access flag for FDCAN2
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 0 **FDCAN1F**: illegal access flag for FDCAN1 and FDCAN message RAM
0: no illegal access event
1: illegal access event

5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3)

Address offset: 0x018

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCBF	DAC10F	DAC9F	DAC8F	RAMCFGF	Res.	XSPI1F	DAC7F	DAC6F	DAC5F	PKAF	SAESF	RNGF	HASHF	AESF	ADC3F
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC12F	DCACHEF	ICACHEF	FMAC2F	FMAC1F	CORDIC1F	CRCF	POSENC2F	POSENC1F	TIM27F	VREFBUFF	COMP8F	COMP7F	COMP6F	COMP5F	COMP4F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **CCBF:** illegal access flag for CCB
0: no illegal access event
1: illegal access event

Bit 30 **DAC10F:** illegal access flag for DAC10
0: no illegal access event
1: illegal access event

Bit 29 **DAC9F:** illegal access flag for DAC9
0: no illegal access event
1: illegal access event

Bit 28 **DAC8F:** illegal access flag for DAC8
0: no illegal access event
1: illegal access event

Bit 27 **RAMCFGF:** illegal access flag for RAMCFG
0: no illegal access event
1: illegal access event

Bit 26 Reserved, must be kept at reset value.

Bit 25 **XSPI1F:** illegal access flag for XSPI1
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 24 **DAC7F:** illegal access flag for DAC7
0: no illegal access event
1: illegal access event

Bit 23 **DAC6F:** illegal access flag for DAC6
0: no illegal access event
1: illegal access event

Bit 22 **DAC5F:** illegal access flag for DAC5
0: no illegal access event
1: illegal access event

Bit 21 **PKAF:** illegal access flag for PKA
0: no illegal access event
1: illegal access event

Bit 20 **SAESF:** illegal access flag for SAES
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 19 **RNGF:** illegal access flag for RNG
0: no illegal access event
1: illegal access event

Bit 18 **HASHF:** illegal access flag for HASH
0: no illegal access event
1: illegal access event

Bit 17 **AESF:** illegal access flag for AES
0: no illegal access event
1: illegal access event

Bit 16 **ADC3F:** illegal access flag for ADC3
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 15 **ADC12F:** illegal access flag for ADC1 and ADC2
0: no illegal access event
1: illegal access event

Bit 14 **DCACHEF:** illegal access flag for DCACHE
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 13 **ICACHEF:** illegal access flag for ICACHE
0: no illegal access event
1: illegal access event

Bit 12 **FMAC2F:** illegal access flag for register for FMAC2
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 11 **FMAC1F:** illegal access flag for register for FMAC1
0: no illegal access event
1: illegal access event

Bit 10 **CORDIC1F**: illegal access flag for CORDIC1
0: no illegal access event
1: illegal access event

Bit 9 **CRCF**: illegal access flag for CRC
0: no illegal access event
1: illegal access event

Bit 8 **POSENC2F**: illegal access flag for POSENC2
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 7 **POSENC1F**: illegal access flag for POSENC1
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 6 **TIM27F**: illegal access flag for TIM27
0: no illegal access event
1: illegal access event

Bit 5 **VREFBUFF**: illegal access flag for VREFBUF
0: no illegal access event
1: illegal access event

Bit 4 **COMP8F**: illegal access flag for COMP8
0: no illegal access event
1: illegal access event

Bit 3 **COMP7F**: illegal access flag for COMP7
0: no illegal access event
1: illegal access event

Bit 2 **COMP6F**: illegal access flag for COMP6
0: no illegal access event
1: illegal access event

Bit 1 **COMP5F**: illegal access flag for COMP5
0: no illegal access event
1: illegal access event

Bit 0 **COMP4F**: illegal access flag for COMP4
0: no illegal access event
1: illegal access event

5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4)

Address offset: 0x01C

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	MPCBB3_REGF	SRAM3F	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	Res.	Res.	Res.	Res.	Res.	XSPI1_MEMF	TZIC1F	TZSC1F
		r	r	r	r	r	r						r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PLAY1F	EXTIF	RCCF	PWRF	TAMPF	RTCF	SBSF	Res.	OTFDEC1F	FLASH_REGF	FLASHF	GPDMA2F	GPDMA1F
			r	r	r	r	r	r	r		r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **MPCBB3_REGF:** illegal access flag for MPCBB3 registers
0: no illegal access event
1: illegal access event
Note: depending on device.

Bit 28 **SRAM3F:** illegal access flag for SRAM3
0: no illegal access event
1: illegal access event
Note: depending on device.

Bit 27 **MPCBB2_REGF:** illegal access flag for MPCBB2 registers
0: no illegal access event
1: illegal access event

Bit 26 **SRAM2F:** illegal access flag for SRAM2
0: no illegal access event
1: illegal access event

Bit 25 **MPCBB1_REGF:** illegal access flag for MPCBB1 registers
0: no illegal access event
1: illegal access event

Bit 24 **SRAM1F:** illegal access flag for SRAM1
0: no illegal access event
1: illegal access event

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **XSPI1_MEMF:** illegal access flag for MPCWM1 (XSPI1) memory bank
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 17 **TZIC1F:** illegal access flag for GTZC1 TZIC registers
0: no illegal access event
1: illegal access event

Bit 16 **TZSC1F:** illegal access flag for GTZC1 TZSC registers
0: no illegal access event
1: illegal access event

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLAY1F**: illegal access flag for PLAY1
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 11 **EXTIF:** illegal access flag for EXTI
0: no illegal access event
1: illegal access event

Bit 10 **RCCF:** illegal access flag for RCC
0: no illegal access event
1: illegal access event

Bit 9 **PWRF:** illegal access flag for PWR
0: no illegal access event
1: illegal access event

Bit 8 **TAMPF:** illegal access flag for TAMP
0: no illegal access event
1: illegal access event

Bit 7 **RTCF:** illegal access flag for RTC
0: no illegal access event
1: illegal access event

Bit 6 **SBSF:** illegal access flag for SBS
0: no illegal access event
1: illegal access event

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC1F:** illegal access flag for OTFDEC1
0: no illegal access event
1: illegal access event
Note: depending on device

Bit 3 **FLASH_REGF:** illegal access flag for FLASH registers
0: no illegal access event
1: illegal access event

Bit 2 **FLASHF:** illegal access flag for FLASH memory
0: no illegal access event
1: illegal access event

Bit 1 **GPDMA2F:** illegal access flag for GPDMA2
0: no illegal access event
1: illegal access event

Bit 0 **GPDMA1F:** illegal access flag for GPDMA1
0: no illegal access event
1: illegal access event

5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1)

Address offset: 0x020

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DTSCF	Res.	TIM11CF	TIM10CF	TIM9CF	DAC2CF	DAC1CF	Res.	Res.	UART6CF	CRSCF	I3C1CF	I2C2CF	I2C1CF	UART5CF
	w		w	w	w	w	w			w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4CF	USART3CF	USART2CF	SPI3CF	SPI2CF	IWDGCF	WWDGCF	TIM14CF	TIM13CF	TIM12CF	TIM7CF	TIM6CF	TIM5CF	TIM4CF	TIM3CF	TIM2CF
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DTSCF:** clear the illegal access flag for DTS
0: no action
1: status flag cleared

Bit 29 Reserved, must be kept at reset value.

Bit 28 **TIM11CF:** clear the illegal access flag for TIM11
0: no action
1: status flag cleared

Bit 27 **TIM10CF:** clear the illegal access flag for TIM10
0: no action
1: status flag cleared

Bit 26 **TIM9CF:** clear the illegal access flag for TIM9
0: no action
1: status flag cleared

Bit 25 **DAC2CF:** clear the illegal access flag for DAC2
0: no action
1: status flag cleared

Bit 24 **DAC1CF:** clear the illegal access flag for DAC1
0: no action
1: status flag cleared

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **UART6CF:** clear the illegal access flag for UART6
0: no action
1: status flag cleared

Bit 20 **CRSCF:** clear the illegal access flag for CRS
0: no action
1: status flag cleared
Note: depending on device

Bit 19 **I3C1CF:** clear the illegal access flag for I3C1
0: no action
1: status flag cleared

Bit 18 **I2C2CF:** clear the illegal access flag for I2C2
0: no action
1: status flag cleared

Bit 17 **I2C1CF:** clear the illegal access flag for I2C1
0: no action
1: status flag cleared

Bit 16 **UART5CF:** clear the illegal access flag for UART5
0: no action
1: status flag cleared

Bit 15 **UART4CF:** clear the illegal access flag for UART4
0: no action
1: status flag cleared

Bit 14 **USART3CF:** clear the illegal access flag for USART3
0: no action
1: status flag cleared
Note: depending on device

Bit 13 **USART2CF:** clear the illegal access flag for USART2
0: no action
1: status flag cleared

Bit 12 **SPI3CF:** clear the illegal access flag for SPI3
0: no action
1: status flag cleared
Note: depending on device

Bit 11 **SPI2CF:** clear the illegal access flag for SPI2
0: no action
1: status flag cleared

Bit 10 **IWDGCF:** clear the illegal access flag for IWDG
0: no action
1: status flag cleared

Bit 9 **WWDGCF:** clear the illegal access flag for WWDG
0: no action
1: status flag cleared

Bit 8 **TIM14CF:** clear the illegal access flag for TIM14
0: no action
1: status flag cleared

Bit 7 **TIM13CF:** clear the illegal access flag for TIM13
0: no action
1: status flag cleared

Bit 6 **TIM12CF:** clear the illegal access flag for TIM12
0: no action
1: status flag cleared

Bit 5 **TIM7CF:** clear the illegal access flag for TIM7
0: no action
1: status flag cleared

Bit 4 **TIM6CF:** clear the illegal access flag for TIM6
0: no action
1: status flag cleared

Bit 3 **TIM5CF:** clear the illegal access flag for TIM5
0: no action
1: status flag cleared
Note: depending on device

Bit 2 **TIM4CF:** clear the illegal access flag for TIM4
0: no action
1: status flag cleared
Note: depending on device

Bit 1 **TIM3CF:** clear the illegal access flag for TIM3
0: no action
1: status flag cleared
Note: depending on device

Bit 0 **TIM2CF:** clear the illegal access flag for TIM2
0: no action
1: status flag cleared

5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP 3CF	COMP 2CF	COMP 1CF	LPTIM1 CF	I2C4CF	I2C3CF	LPUAR T1CF	TIM26 CF	OPAMP 3CF	OPAMP 2CF	OPAMP 1CF	TIM20 CF	USBCF	HRTIM 1CF	Res.	Res.
w	w	w	w	w	w	w	w	w	w	w	w	w	w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TIM17 CF	TIM16 CF	TIM15 CF	USART 1CF	TIM8C F	SPI1CF	TIM1C F	Res.	Res.	Res.	Res.	UCPD1 CF	FDCAN 3CF	FDCAN 2CF	FDCAN 1CF
	w	w	w	w	w	w	w					w	w	w	w

Bit 31 **COMP3CF:** clear the illegal access flag for COMP3
0: no action
1: status flag cleared

Bit 30 **COMP2CF:** clear the illegal access flag for COMP2
0: no action
1: status flag cleared

Bit 29 **COMP1CF:** clear the illegal access flag for COMP1
0: no action
1: status flag cleared

Bit 28 **LPTIM1CF:** clear the illegal access flag for LPTIM1
0: no action
1: status flag cleared

Bit 27 **I2C4CF:** clear the illegal access flag for I2C4
0: no action
1: status flag cleared
Note: depending on device

Bit 26 **I2C3CF:** clear the illegal access flag for I2C3
0: no action
1: status flag cleared
Note: depending on device

Bit 25 **LPUART1CF:** clear the illegal access flag for LPUART1
0: no action
1: status flag cleared

Bit 24 **TIM26CF:** clear the illegal access flag for TIM26
0: no action
1: status flag cleared

Bit 23 **OPAMP3CF:** clear the illegal access flag for OPAMP3
0: no action
1: status flag cleared

Bit 22 **OPAMP2CF:** clear the illegal access flag for OPAMP2
0: no action
1: status flag cleared

Bit 21 **OPAMP1CF:** clear the illegal access flag for OPAMP1
0: no action
1: status flag cleared

Bit 20 **TIM20CF:** clear the illegal access flag for TIM20
0: no action
1: status flag cleared
Note: depending on device

Bit 19 **USBCF:** clear the illegal access flag for USB
0: no action
1: status flag cleared
Note: depending on device

Bit 18 **HRTIM1CF:** clear the illegal access flag for HRTIM1
0: no action
1: status flag cleared

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **TIM17CF:** clear the illegal access flag for TIM17
0: no action
1: status flag cleared

Bit 13 **TIM16CF:** clear the illegal access flag for TIM16
0: no action
1: status flag cleared

Bit 12 **TIM15CF:** clear the illegal access flag for TIM15
0: no action
1: status flag cleared

Bit 11 **USART1CF:** clear the illegal access flag for USART1
0: no action
1: status flag cleared

Bit 10 **TIM8CF:** clear the illegal access flag for TIM8
0: no action
1: status flag cleared

Bit 9 **SPI1CF:** clear the illegal access flag for SPI1
0: no action
1: status flag cleared

Bit 8 **TIM1CF:** clear the illegal access flag for TIM1
0: no action
1: status flag cleared

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **UCPD1CF**: clear the illegal access flag for UCPD1
0: no action
1: status flag cleared
Note: depending on device

Bit 2 **FDCAN3CF**: clear the illegal access flag for FDCAN3
0: no action
1: status flag cleared
Note: depending on device

Bit 1 **FDCAN2CF**: clear the illegal access flag for FDCAN2
0: no action
1: status flag cleared
Note: depending on device

Bit 0 **FDCAN1CF**: clear the illegal access flag for FDCAN1 and FDCAN message RAM
0: no action
1: status flag cleared

5.7.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3)

Address offset: 0x028

Reset value: 0x0000 0000

Secure privilege access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCBCF	DAC10CF	DAC9CF	DAC8CF	RAMCFGCF	Res.	XSPI1CF	DAC7CF	DAC6CF	DAC5CF	PKACF	SAESCF	RNGCF	HASHCF	AESCF	ADC3CF
w	w	w	w	w		w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC12CF	DCACHECF	ICACHECF	FMAC2CF	FMAC1CF	CORDIC1CF	CRCCF	POSENC2CF	POSENC1CF	TIM27CF	VREFBUFCF	COMP8CF	COMP7CF	COMP6CF	COMP5CF	COMP4CF
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **CCBCF:** clear the illegal access flag for CCB
0: no action
1: status flag cleared

Bit 30 **DAC10CF:** clear the illegal access flag for DAC10
0: no action
1: status flag cleared

Bit 29 **DAC9CF:** clear the illegal access flag for DAC9
0: no action
1: status flag cleared

Bit 28 **DAC8CF:** clear the illegal access flag for DAC8
0: no action
1: status flag cleared

Bit 27 **RAMCFGCF:** clear the illegal access flag for RAMCFG
0: no action
1: status flag cleared

Bit 26 Reserved, must be kept at reset value.

Bit 25 **XSPI1CF:** clear the illegal access flag for XSPI1
0: no action
1: status flag cleared
Note: depending on device

Bit 24 **DAC7CF:** clear the illegal access flag for DAC7
0: no action
1: status flag cleared

Bit 23 **DAC6CF:** clear the illegal access flag for DAC6
0: no action
1: status flag cleared

Bit 22 **DAC5CF:** clear the illegal access flag for DAC5
0: no action
1: status flag cleared

Bit 21 **PKACF:** clear the illegal access flag for PKA
0: no action
1: status flag cleared

Bit 20 **SAESCF:** clear the illegal access flag for SAES
0: no action
1: status flag cleared
Note: depending on device

Bit 19 **RNGCF:** clear the illegal access flag for RNG
0: no action
1: status flag cleared

Bit 18 **HASHCF:** clear the illegal access flag for HASH
0: no action
1: status flag cleared

Bit 17 **AESCF:** clear the illegal access flag for AES
0: no action
1: status flag cleared

Bit 16 **ADC3CF:** clear the illegal access flag for ADC3
0: no action
1: status flag cleared
Note: depending on device

Bit 15 **ADC12CF:** clear the illegal access flag for ADC1 and ADC2
0: no action
1: status flag cleared

Bit 14 **DCACHECF:** clear the illegal access flag for DCACHE
0: no action
1: status flag cleared
Note: depending on device

Bit 13 **ICACHECF:** clear the illegal access flag for ICACHE
0: no action
1: status flag cleared

Bit 12 **FMAC2CF:** clear the illegal access flag for register of FMAC2
0: no action
1: status flag cleared
Note: depending on device

Bit 11 **FMAC1CF:** clear the illegal access flag for register of FMAC1
0: no action
1: status flag cleared

Bit 10 **CORDIC1CF:** clear the illegal access flag for CORDIC1
0: no action
1: status flag cleared

Bit 9 **CRCCF:** clear the illegal access flag for CRC
0: no action
1: status flag cleared

Bit 8 **POSENC2CF:** clear the illegal access flag for POSENC2
0: no action
1: status flag cleared
Note: depending on device

Bit 7 **POSENC1CF:** clear the illegal access flag for POSENC1
0: no action
1: status flag cleared
Note: depending on device

Bit 6 **TIM27CF:** clear the illegal access flag for TIM27
0: no action
1: status flag cleared

Bit 5 **VREFBUFCF:** clear the illegal access flag for VREFBUF
0: no action
1: status flag cleared

Bit 4 **COMP8CF:** clear the illegal access flag for COMP8
0: no action
1: status flag cleared

Bit 3 **COMP7CF:** clear the illegal access flag for COMP7
0: no action
1: status flag cleared

Bit 2 **COMP6CF:** clear the illegal access flag for COMP6
0: no action
1: status flag cleared

Bit 1 **COMP5CF:** clear the illegal access flag for COMP5
0: no action
1: status flag cleared

Bit 0 **COMP4CF:** clear the illegal access flag for COMP4
0: no action
1: status flag cleared

5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4)

Address offset: 0x02C

Reset value: 0x0000 0000

Secure privilege access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	MPCBB3_REGCF	SRAM3CF	MPCBB2_REGCF	SRAM2CF	MPCBB1_REGCF	SRAM1CF	Res.	Res.	Res.	Res.	Res.	XSPI1_MEMCF	TZIC1CF	TZSC1CF
		w	w	w	w	w	w						w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PLAY1CF	EXTICF	RCCCF	PWRCF	TAMPCF	RTCCF	SBSCF	Res.	OTFDEC1CF	FLASH_REGCF	FLASHCF	GPDMA2CF	GPDMA1CF
			w	w	w	w	w	w	w		w	w	w	w	w

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **MPCBB3_REGCF:** clear the illegal access flag for MPCBB3 registers
0: no action
1: status flag cleared
Note: depending on device.

Bit 28 **SRAM3CF:** clear the illegal access flag for SRAM3
0: no action
1: status flag cleared
Note: depending on device.

Bit 27 **MPCBB2_REGCF:** clear the illegal access flag for MPCBB2 registers
0: no action
1: status flag cleared

Bit 26 **SRAM2CF:** clear the illegal access flag for SRAM2
0: no action
1: status flag cleared

Bit 25 **MPCBB1_REGCF:** clear the illegal access flag for MPCBB1 registers
0: no action
1: status flag cleared

Bit 24 **SRAM1CF:** clear the illegal access flag for SRAM1
0: no action
1: status flag cleared

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **XSPI1_MEMCF:** clear the illegal access flag for MPCWM1 (XSPI1) memory bank
0: no action
1: status flag cleared
Note: depending on device

Bit 17 **TZIC1CF:** clear the illegal access flag for GTZC1 TZIC registers
0: no action
1: status flag cleared

Bit 16 **TZSC1CF:** clear the illegal access flag for GTZC1 TZSC registers
0: no action
1: status flag cleared

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PLAY1CF:** clear the illegal access flag for PLAY1
0: no action
1: status flag cleared
Note: depending on device

Bit 11 **EXTICF:** clear the illegal access flag for EXTI
0: no action
1: status flag cleared

Bit 10 **RCCCF:** clear the illegal access flag for RCC
0: no action
1: status flag cleared

Bit 9 **PWRCF:** clear the illegal access flag for PWR
0: no action
1: status flag cleared

Bit 8 **TAMPCF:** clear the illegal access flag for TAMP
0: no action
1: status flag cleared

Bit 7 **RTCCF:** clear the illegal access flag for RTC
0: no action
1: status flag cleared

Bit 6 **SBSCF:** clear the illegal access flag for SBS
0: no action
1: status flag cleared

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC1CF:** clear the illegal access flag for OTFDEC1
0: no action
1: status flag cleared
Note: depending on device

Bit 3 **FLASH_REGCF:** clear the illegal access flag for FLASH memory
0: no action
1: status flag cleared

Bit 2 **FLASHCF:** clear the illegal access flag for FLASH registers
0: no action
1: status flag cleared

Bit 1 **GPDMA2CF:** clear the illegal access flag for GPDMA2
0: no action
1: status flag cleared

Bit 0 **GPDMA1CF:** clear the illegal access flag for GPDMA1
0: no action
1: status flag cleared

5.7.13 GTZC1 TZIC register map

Table 33. GTZC1 TZIC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_TZIC_IER1	Res.	DTSIE	Res.	TIM11IE	TIM10IE	TIM9IE	DAC2IE	DAC1IE	Res.	Res.	UART6IE	CRSIE	I3C1IE	I2C2IE	I2C1IE	UART5IE	UART4IE	USART3IE	USART2IE	SPI3IE	SPI2IE	IWDGIE	WWDGIE	TIM14IE	TIM13IE	TIM12IE	TIM7IE	TIM6IE	TIM5IE	TIM4IE	TIM3IE	TIM2IE
	Reset value		0		0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	GTZC1_TZIC_IER2	COMP3IE	COMP2IE	COMP1IE	LPTIM1IE	I2C4IE	I2C3IE	LPUART1IE	TIM26IE	OPAMP3IE	OPAMP2IE	OPAMP1IE	TIM20IE	USBIE	HRTIM1IE				TIM17IE	TIM16IE	TIM15IE	USART1IE	TIM8IE	SPI1IE	TIM1IE					UCPD1IE	FDCAN3IE	FDCAN2IE	FDCAN1IE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0					0	0	0	0
0x008	GTZC1_TZIC_IER3	CCBIE	DAC10IE	DAC9IE	DAC8IE	RAMCFGIE	Res.	XSPI1IE	DAC7IE	DAC6IE	DAC5IE	PKAIE	SAESIE	RNGIE	HASHIE	AESIE	ADC3IE	ADC12IE	DCACHEIE	ICACHEIE	FMAC2IE	FMAC1IE	CORDIC1IE	CRCIE	POSENC2IE	POSENC1IE	TIM27IE	VREFBUFIE	COMP8IE	COMP7IE	COMP6IE	COMP5IE	COMP4IE
	Reset value	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	GTZC1_TZIC_IER4	Res.	Res.	MPCBB3_REGIE	SRAM3IE	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	Res.	Res.	Res.	Res.	Res.	XSPI1_MEMIE	TZIC1IE	TZSC1IE	Res.	Res.	Res.	PLAY1IE	EXTIIE	RCCIE	PWRIE	TAMPIE	RTCIE	SBSIE	Res.	OTFDEC1IE	FLASH_REGIE	FLASHIE	GPDMA2IE	GPDMA1IE
	Reset value			0	0	0	0	0	0						0	0	0				0	0	0	0	0	0	0		0	0	0	0	0
0x010	GTZC1_TZIC_SR1	Res.	DTSF	Res.	TIM11F	TIM10F	TIM9F	DAC2F	DAC1F	Res.	Res.	USART6F	CRSF	I3C1F	I2C2F	I2C1F	UART5F	UART4F	USART3F	USART2F	SPI3F	SPI2F	IWDGF	WWDGF	TIM14F	TIM13F	TIM12F	TIM7F	TIM6F	TIM5F	TIM4F	TIM3F	TIM2F
	Reset value		0		0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 33. GTZC1 TZIC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	GTZC1_TZIC_SR2	COMP3F	COMP2F	COMP1F	LPTIM1F	I2C4F	I2C3F	LPUART1F	TIM26F	OPAMP3F	OPAMP2F	OPAMP1F	TIM20F	USBF	HRTIM1F		Res.	Res.	TIM17F	TIM16F	TIM15F	USART1F	TIM8F	SPI1F	TIM1F	Res.	Res.	Res.		UCPD1F	FDCAN3F	FDCAN2F	FDCAN1F
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0					0	0	0	0
0x018	GTZC1_TZIC_SR3	CCBF	DAC10F	DAC9F	DAC8F	RAMCFGF	Res.	XSPI1F	DAC7F	DAC6F	DAC5F	PKAF	SAESF	RNGF	HASHF	AESF	ADC3F	ADC12F	DCACHEF	ICACHEF	FMAC2F	FMAC1F	CORDIC1F	CRCF	POSENC2F	POSENC1F	TIM27F	VREFBUFF	COMP8F	COMP7F	COMP6F	COMP5F	COMP4F
	Reset value	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	GTZC1_TZIC_SR4	Res.	Res.	MPCBB3_REGF	SRAM3F	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	Res.	Res.	Res.	Res.	Res.	XSPI1_MEMF	TZIC1F	TZSC1F	Res.	Res.	Res.	PLAY1F	EXTIF	RCCF	PWRF	TAMPF	RTCF	SBSF	Res.	OTFDEC1F	FLASH_REGF	FLASHF	GPDMA2F	GPDMA1F
	Reset value			0	0	0	0	0	0						0	0	0				0	0	0	0	0	0	0		0	0	0	0	0
0x020	GTZC1_TZIC_FCR1	Res.	DTSCF	Res.	TIM11CF	TIM10CF	TIM9CF	DAC2CF	DAC1CF	Res.	Res.	UART6CF	CRSCF	I3C1CF	I2C2CF	I2C1CF	UART5CF	UART4CF	USART3CF	USART2CF	SPI3CF	SPI2CF	IWDGCF	WWDGCF	TIM14CF	TIM13CF	TIM12CF	TIM7CF	TIM6CF	TIM5CF	TIM5CF	TIM3CF	TIM2CF
	Reset value		0		0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x024	GTZC1_TZIC_FCR2	COMP3CF	COMP2CF	COMP1CF	LPTIM1CF	I2C4CF	I2C3CF	LPUART1CF	TIM26CF	OPAMP3CF	OPAMP2CF	OPAMP1CF	TIM20CF	USBCF	HRTIM1CF		Res.	Res.	TIM17CF	TIM16CF	TIM15CF	USART1CF	TIM8CF	SPI1CF	TIM1CF	Res.	Res.	Res.	Res.	UCPD1CF	FDCAN3CF	FDCAN2CF	FDCAN1CF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0					0	0	0	0
0x028	GTZC1_TZIC_FCR3	CCBCF	DAC10CF	DAC9CF	DAC8CF	RAMCFGCF	Res.	XSPI1CF	DAC7CF	DAC6CF	DAC5CF	PKACF	SAESCF	RNGCF	HASHCF	AESCF	ADC3CF	ADC12CF	DCACHECF	ICACHECF	FMAC2CF	FMAC1CF	CORDIC1CF	CRCCF	POSENC2CF	POSENC1CF	TIM27CF	VREFBUFCF	COMP8CF	COMP7CF	COMP6CF	COMP5CF	COMP4CF
	Reset value	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	GTZC1_TZIC_FCR4	Res.	Res.	MPCBB3_REGCF	SRAM3CF	MPCBB2_REGCF	SRAM2CF	MPCBB1_REGCF	SRAM1CF	Res.	Res.	Res.	Res.	Res.	XSPI1_MEMCF	TZIC1CF	TZSC1CF	Res.	Res.	Res.	PLAY1CF	EXTICF	RCCCF	PWRCF	TAMPCF	RTCCF	SBSCF	Res.	OTFDEC1CF	FLASH_REGCF	FLASHCF	GPDMA2CF	GPDMA1CF
	Reset value			0	0	0	0	0	0				0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0

Refer to *Table 25: GTZC1 subblock address offset*.

5.8 GTZC1 MPCBBz registers (z = 1 to 3)

All registers are accessed only by words (32-bit).

5.8.1 GTZC1 SRAMz MPCBB control register (GTZC1_MPCBBz_CR) (z = 1 to 3)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bit 31 **SRWILADIS**: secure read/write illegal access disable

This bit disables the detection of an illegal access when a secure read/write transaction access a nonsecure blocks of the block-based SRAM (secure fetch on nonsecure block is always considered illegal).

0: enabled, secure read/write access not allowed on nonsecure SRAM block

1: disabled, secure read/write access allowed on nonsecure SRAM block

Bit 30 **INVSECSTATE**: SRAMx clocks security state

This bit is used to define the internal SRAMs clocks control in RCC as secure or not.

0: SRAMs clocks are secure if a secure area exists in the MPCBB. It is non secure if there is no secure area.

1: SRAMs clocks are nonsecure even if a secure area exists in the MPCBB, and secure even if no secure block is set in the MPCBB.

Bits 29:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: lock the control register of the MPCBB until next reset

This bit is cleared by default, and once set, it cannot be reset until system reset.

0: control register not locked

1: control register locked

5.8.2 GTZC1 SRAMz MPCBB configuration lock register 1 (GTZC1_MPCBBz_CFGLOCK1) (z = 1 to 3)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLCK 31	SPLCK 30	SPLCK 29	SPLCK 28	SPLCK 27	SPLCK 26	SPLCK 25	SPLCK 24	SPLCK 23	SPLCK 22	SPLCK 21	SPLCK 20	SPLCK 19	SPLCK 18	SPLCK 17	SPLCK 16
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPLCK 15	SPLCK 14	SPLCK 13	SPLCK 12	SPLCK 11	SPLCK 10	SPLCK 9	SPLCK 8	SPLCK 7	SPLCK 6	SPLCK 5	SPLCK 4	SPLCK 3	SPLCK 2	SPLCK 1	SPLCK 0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SPLCKy**: Security/privilege configuration lock for super-block (y = 31 to 0)
This bit is set by software and can be cleared only by system reset.
0: GTZC1_MPCBBz_SECCFGRy and GTZC1_MPCBBz_PRIVCFGRy can be written.
1: Writes to GTZC1_MPCBBz_SECCFGRy and GTZC1_MPCBBz_PRIVCFGRy are ignored

5.8.3 GTZC1 SRAMz MPCBB security configuration for super-block x register (GTZC1_MPCBBz_SECCFGRx) (z = 1 to 3)

Address offset: 0x100 + 0x4 * x, (x = 0 to 31)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0000 0000 when TZEN = 0xC3.

Write access to this register is secure only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECy**: Security configuration for block y (y = 31 to 0)
0: nonsecure access only to block y, belonging to super-block x. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBBz_CR.
1: Secure access only to block y, belonging to super-block x.
Unprivileged write to this bit is ignored if PRIVy bit is set in GTZC1_MPCBBz_PRIVCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBBz_CFGLOCK.

5.8.4 GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 3)

Address offset: 0x200 + 0x4 * x, (x = 0 to 31)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0000 0000 when TZEN = 0xC3. Write access to this register is privileged only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVy**: Privileged configuration for block y, belonging to super-block x (y = 31 to 0).
0: Privileged and unprivileged access to block y, belonging to super-block x
1: Only privileged access to block y, belonging to super-block x
nonsecure write to this bit is ignored if SECy bit is set in GTZC1_MPCBBz_SECCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBBz_CFGLOCK.

5.8.5 GTZC1 MPCBBz register map (z = 1 to 3)

Table 34. GTZC1 MPCBBz register map and reset values (z = 1 to 3)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_ MPCBBz_CR	SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value	0	0																														0
0x004-0x00C	Reserved	Reserved																															
0x010	**GTZC1_MPCBBz_ CFGLOCK1**	SPLCK31	SPLCK30	SPLCK29	SPLCK28	SPLCK27	SPLCK26	SPLCK25	SPLCK24	SPLCK23	SPLCK22	SPLCK21	SPLCK20	SPLCK19	SPLCK18	SPLCK17	SPLCK16	SPLCK15	SPLCK14	SPLCK13	SPLCK12	SPLCK11	SPLCK10	SPLCK9	SPLCK8	SPLCK7	SPLCK6	SPLCK5	SPLCK4	SPLCK3	SPLCK2	SPLCK1	SPLCK0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014-0x0FC	Reserved	Reserved																															
0x100 + 0x04 *x (x = 0 to 31)	**GTZC1_MPCBBz_ SECCFGRx**	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x180-0x1FC	Reserved	Reserved																															
0x200 + 0x04 *x (x = 0 to 31)	**GTZC1_MPCBBz_ PRIVCFGRx**	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Table 25: GTZC1 subblock address offset*.

6 RAMs configuration controller (RAMCFG)

6.1 Introduction

The RAMCFG configures the features of the internal SRAMs (SRAM1, SRAM2 and SRAM3).

Note: SRAM3 resource is supported by STM32G55x/56x, not implemented on STM32G53x/54x

6.2 RAMCFG main features

The internal SRAM supports some of the features listed hereafter, configured in RAMCFG:

- Error code correction (ECC):
  - Single error detection and correction with interrupt generation
  - Double error detection with interrupt or NMI generation
  - Status with failing address
- Write protection (1-Kbyte granularity)
- SRAM software erase

6.3 RAMCFG functional description

6.3.1 Internal SRAMs features

Five SRAMs are embedded in the devices, each with specific features:

- SRAM1, SRAM2, SRAM3 are the main SRAMs, made of several blocks that can be powered down in Stop mode to reduce consumption.
- Block partitioning for STM32G53x/54x:
  - SRAM1: 16-Kbyte + 4-Kbyte blocks (total 20Kbytes)
  - SRAM2: Two 16-Kbyte blocks + 8-Kbyte (total 40Kbytes)
- Block partitioning for STM32G55x/56x:
  - SRAM1: one 32-Kbyte block
  - SRAM2: 16-Kbyte + 48-Kbyte blocks (total 64Kbytes)
  - SRAM3: one 16-Kbyte block
- SRAM2 is erased when a system reset occurs if the SRAM2_RST option bit is selected in the flash memory user option bytes. SRAM1 and SRAM3 are erased when a system reset occurs if the SRAM13_RST option bit is selected in the flash memory user option bytes. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.
- SRAM1 and SRAM2 are protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. Refer to *Section 50: Tamper and backup registers (TAMP)* for more details.
- The RAMCFG embeds the registers related to the internal SRAMs ECC, write protection and software erase.

Table 35 summarizes the features supported by each internal SRAM.

Table 35. Internal SRAMs features

SRAM feature	SRAM1	SRAM2	SRAM3
Size for STM32G53x/54x (Kbytes)	20	40	-
Size for STM32G55x/56x (Kbytes)	32	64	16
Optional retention in Standby mode	-	-	-
Erased with tamp.er detection and backup domain reset	X	X	-
Optionally erased with system reset	X	X	X
Software erase	X	X	X
ECC	X	X	X
Write protection	X	X	-

6.3.2 Error code correction (SRAM1, SRAM2, SRAM3)

The ECC is supported by SRAM1, SRAM2 and SRAM3 when enabled with the SRAM1_ECC, SRAM2_ECC, SRAM3_ECC user option bits. Refer to*Section 7: Embedded flash memory (FLASH)* for more details.

Seven ECC bits are added per 32 bits of SRAM, allowing two bits error detection and one bit error correction on memory read access.

As the ECC is calculated and checked for a 32-bit word, the byte and half-word write accesses are managed by the SRAM interface by first reading the whole word, then write the word again with the new byte/half-word value. ECC double errors are also detected during these byte or half-word AHB write accesses (read/modify/write done by interface). The byte or half-word write access latency is two AHB clock cycles.

Caution: In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM two AHB clock cycles after the SRAM AHB is released (when SRAM is no more accessed).

Single and double ECC errors

When a single error is detected, it is automatically corrected and the SEDC/CSEDC bits are set, respectively, in RAMCFG_MxISR/RAMCFG_MxICR. An interrupt is generated if enabled by the SEIE bit in RAMCFG_MxIER. The failing address is stored in RAMCFG_MxSEAR if the ALE bit is set in RAMCFG_MxCR.

Caution: Single errors cannot be detected when the SEDC bit is set.

When a double error is detected, the DED and CDED bits are set in RAMCFG_MxISR and RAMCFG_MxICR respectively. An interrupt or NMI is generated if enabled by the DEIE or ECCNMI bit in RAMCFG_MxIER. The failing address is stored in RAMCFG_MxDEAR if the ALE bit is set in RAMCFG_MxCR.

Caution: Double errors cannot be detected when the DED bit is set.

When ECC is enabled by user option bits, the ECCE bit is automatically set after system reset in related RAMCFG_MxCR.

The ECC can be deactivated by executing the following software sequence:

1. Write 0xAE in RAMCFG_MxECCKEYR.
2. Write 0x75 in RAMCFG_MxECCKEYR.
3. Write 0 in RAMCFG_MxCR.

6.3.3 Write protection

The SRAM1 and SRAM2 is made of 1-Kbyte pages, each of them can be write-protected by setting its corresponding PyWP bit (0<y<max. page number) in RAMCFG_MxWPR1 and RAMCFG_MxWPR2 (x = 1, 2).

6.3.4 Software erase

SRAM erase can be requested by executing this software sequence:

1. Write 0xCA in RAMCFG_MxERKEYR.
2. Write 0x53 in RAMCFG_MxERKEYR.
3. Write 1 in RAMCFG_MxCR.

SRAMBUSY flag is set in the related SRAM interrupt status register as long as the erase is ongoing.

The total duration of each SRAM erase is N AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is read or written while an erase is ongoing, accesses on the AHB bus are RAZ/WI until the end of the operation.

6.4 RAMCFG low-power modes

Table 36. Effect of low-power modes on RAMCFG

Mode	Description
Sleep	No effect. RAMCFG interrupts cause the device to exit the Sleep mode
Stop	The content of RAMCFG registers is kept
Standby	The RAMCFG is powered down and must be reinitialized after exiting Standby mode.

6.5 RAMCFG interrupts

Table 37. RAMCFG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit(s)	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby modes
RAMCFG	ECC single error detection and correction	SEDC	SEIE	Write 1 in CSEDC	Yes	No	No
	ECC double error detection	DED	DEIE = 1 and ECCNMI = 0	Write 1 in CDED	Yes	No	No
NMI	ECC double error detection	DED	ECCNMI	Write 1 in CDED	Yes	No	No

6.6 RAMCFG registers

In the registers described below, x refers to:

- SRAM1/2/3 when x = 1/2/3, respectively for STM32G55x/56x
- SRAM1/2 when x = 1/2, respectively for STM32G53x/54x

6.6.1 RAMCFG memory x control register (RAMCFG_MxCR)

Address offset: 0x00 + 0x040 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 000X

ECCE reset value depends on ECC enable user option bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
							rs				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the RAMCFG_MxERKEYR register. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation on going

1: Erase operation on going

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: Address latch enable

0: Failing address not stored in the SRAMx ECC single/double error address registers

1: Failing address stored in the SRAMx ECC single/double error address registers

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ECCE**: ECC enable.
This bit reset value is defined by the user option bit configuration. When set, it can be cleared by software only after writing the unlock sequence in the RAMCFG_MxECCKEYR register.
0: ECC disabled
1: ECC enabled

6.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxIER)

Address offset: 0x004 + 0x040 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
												rs		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ECCNMI**: Double error NMI
This bit is set by software and cleared only by a global RAMCFG reset.
0: NMI not generated in case of ECC double error
1: NMI generated in case of ECC double error
Note: if ECCNMI is set, the RAMCFG maskable interrupt is not generated whatever DEIE bit value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DEIE**: ECC double error interrupt enable
0: Double error interrupt disabled
1: Double error interrupt enabled

Bit 0 **SEIE**: ECC single error interrupt enable
0: Single error interrupt disabled
1: Single error interrupt enabled

6.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR)

Address offset: 0x008 + 0x040 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
							r							r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation on going

1: Erase operation on going

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option bit is enabled, tamper detection or readout protection regression. Refer to Table 35.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DED**: ECC double error detected

0: No double error

1: Double error detected

Bit 0 **SEDC**: ECC single error detected and corrected

0: No single error

1: Single error detected and corrected

6.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR)

Address offset: 0x00C + 0x040 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESEA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ESEA[31:0]**: ECC single error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC single error.

6.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR)

Address offset: 0x010 + 0x040 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EDEA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EDEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EDEA[31:0]**: ECC double error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC double error.

6.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)

Address offset: 0x014 + 0x040 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CDED**: Clear ECC double error detected

Writing 1 to this flag clears the DED bit in the RAMCFG_MxISR register. Reading this flag returns the DED value.

Bit 0 **CSEDC**: Clear ECC single error detected and corrected

Writing 1 to this flag clears the SEDC bit in the RAMCFG_MxISR register. Reading this flag returns the SEDC value.

6.6.7 RAMCFG memory 2 write protection register 1 (RAMCFG_MxWPR1)

Address offset: 0x018 + 0x040 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 31 to 0)

These bits are set by software and cleared only by a global RAMCFG reset.

0: Write protection of SRAMx 1-Kbyte page y is disabled.

1: Write protection of SRAMx 1-Kbyte page y is enabled.

6.6.8 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 63 to 32)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.9 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)

Address offset: 0x024 + 0x040 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ECCKEY[7:0]**: ECC write protection key
The following steps are required to unlock the write protection of the ECCE bit in the RAMCFG_MxCR register.
1) Write 0xAE into ECCKEY[7:0].
2) Write 0x75 into ECCKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

6.6.10 RAMCFG memory x erase key register (RAMCFG_MxERKEYR)

Address offset: 0x028 + 0x040 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of the SRAMER bit in the RAMCFG_MxCR register.

1) Write 0xCA into ERASEKEY[7:0].

2) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

6.6.11 RAMCFG register map

Table 38. RAMCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RAMCFG_M1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x04	RAMCFG_M1IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x08	RAMCFG_M1ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x0C	RAMCFG_M1SEAR	ESEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	RAMCFG_M1DEAR	EDEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	RAMCFG_M1ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x18	RAMCFG_M1WPR1	P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP	P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	Reserved	Reserved																															

Table 38. RAMCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	RAMCFG_M1ECCKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RAMCFG_M1ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x2C to 0x3C	Reserved	Reserved																															
0x40	RAMCFG_M2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x44	RAMCFG_M2IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x48	RAMCFG_M2ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x4C	RAMCFG_M2SEAR	ESEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RAMCFG_M2DEAR	EDEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	RAMCFG_M2ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x58	RAMCFG_M2WPR1	P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP	P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	RAMCFG_M2WPR2	P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP	P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	Reserved	Reserved																															
0x64	RAMCFG_M2ECCKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x68	RAMCFG_M2ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x6C to 0x7C	Reserved	Reserved																															
0x80	RAMCFG_M3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				0
0x84	RAMCFG_M3IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0

Table 38. RAMCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x88	RAMCFG_M3ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x8C	RAMCFG_M3SEAR	ESEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	RAMCFG_M3DEAR	EDEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	RAMCFG_M3ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x98 to 0xA0	Reserved	Reserved																															
0xA4	RAMCFG_M3ECCKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xA8	RAMCFG_M3ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xAC to 0xFC	Reserved	Reserved																															

Refer to *Section 2.3* for the register boundary addresses.

7 Embedded flash memory (FLASH)

7.1 Introduction

The embedded flash memory (FLASH) manages the access of any master device to the embedded nonvolatile memory. It implements the read, program, and erase operations, error corrections, as well as various integrity and confidentiality protection mechanisms.

FLASH manages the automatic loading of nonvolatile user option bytes at power-on reset and implements the dynamic update of these options. It also features a high-cycle data area, a one-time-programmable (OTP) area, a secure key storage area (OBKeys), and a read-only area configured by STMicroelectronics during manufacturing.

7.2 FLASH main features

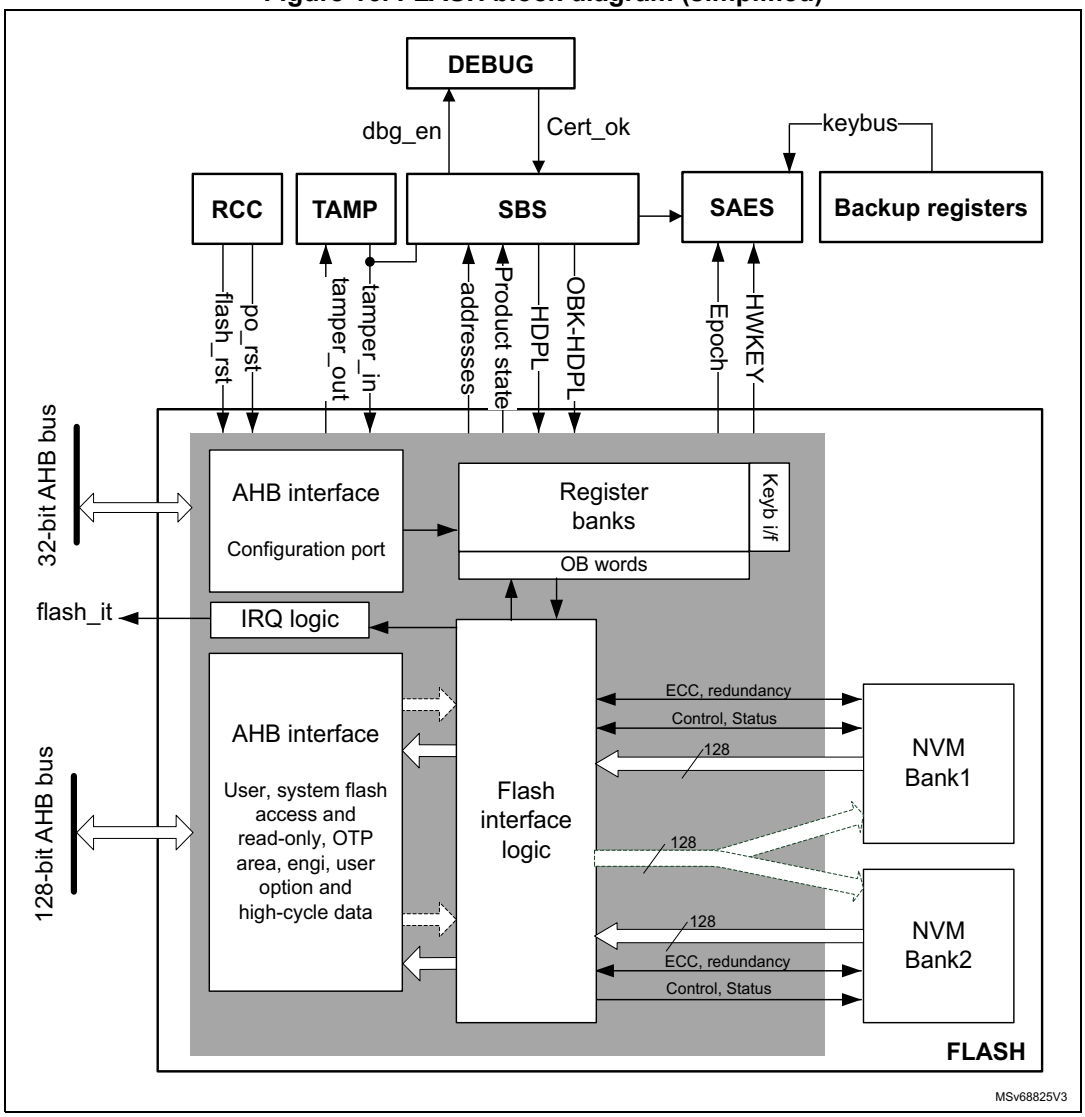
- Up to 512 Kbytes of nonvolatile memory, divided into two equally sized banks (256 Kbytes for STM32G53x/54x and 512 Kbytes for STM32G55x/56x devices)
- Flash memory read operations supporting multiple lengths: 128 bits, 64 bits, 32 bits, 16 bits, or one byte
- Flash memory programming by 128 (user area, OBKeys) and by 16 bits (OTP and flash high-cycle data area)
- 8-Kbyte page erase, bank erase, and dual-bank mass erase
- Dual-bank organization supporting:
  - Simultaneous operations: read-while-write (program and erase) is supported, including flash high-cycle data area. The two banks share the same interface, hence write and erase cannot be performed in parallel (write-while-write is not supported).
  - Bank swapping: the address mapping of the user memory of each bank can be swapped, along with the corresponding registers. Security flags remain valid for the physical bank, so the data are not revealed by swapping to a bank with lower security configuration.
- Error code correction (ECC): one-error detection/correction, or two-error detection per 128-bit flash word using nine ECC bits on the code flash area, on 6-bit words with six ECC bits within the configurable flash high-cycle data area
- User-configurable nonvolatile option bytes
- Flash memory enhanced protections, activated by option bytes
  - Different product states for protecting memory content from debug access
  - Page group write-protection (WRPG), protecting up to 32 groups of four pages (32 Kbytes) per bank
  - Two secure-only areas (one per user flash bank): when enabled, these areas are accessible only if the microcontroller operates in secure access mode
  - HDP protection, providing temporal isolation for startup code
- 2-Kbyte one-time programmable (OTP) area
- Read-only area configured by STMicroelectronics
- Prefetch reads the next sequential instruction from flash memory
- Up to 48 Kbytes per bank (up to 24 Kbytes per bank for STM32G53x/54x and up to 48 Kbytes per bank for STM32G55x/56x devices) supporting high-cycling capability (100 kcycles), to be used for data (EEPROM emulation)

7.3 FLASH functional description

7.3.1 FLASH block diagram

Figure 16 shows the embedded flash memory block diagram.

Figure 16. FLASH block diagram (simplified)



7.3.2 FLASH signals

The flash memory has two AHB connections, namely the flash AHB register interface and the main AHB interface.

Flash AHB register interface

- The data size is 32 bits.
- Except for some registers (FLASH_KEYR/SKEYR, FLASH_OBKKEYR/FLASH_SOBKKEYR, and FLASH_OPTKEYR that only can be written by 32 bits), it is possible to read and write all registers by 8, 16, and 32 bits.
- When the unlock sequence for control and option registers is wrong, a bus error is raised, otherwise no read or write errors are generated on the bus.

Main AHB interface

The AHB data bus size is 128 bits. This interface is used to handle three different targets:

- Code placed in the user and system memory
- Secure keys placed in OBKey pages
- OTP, read-only, and flash memory high-cycle data area

The main AHB interface is implemented as follows:

- User and system memory, OBKeys storage:
  - Supports multiple lengths: 128-, 64-, 32-, 16-, and 8-bit data width.
  - There is a read buffer of 128 bits for each bank where the last data read is stored. If data are available in the read buffer, no read access is given to the flash memory. The buffer is flushed when write access, OTP access, OBK swap, OBK alternate page erase, high-cycle data area access, user option change request, or erase operation occur.
  - There is a prefetch of the same size as the read buffer.
  - A 9-bit ECC complements each 128-bit data flash memory word.
- OTP, read-only, and flash high-cycle data
  - Two dedicated data buffers of 137 bits are used to manage 16-bit data with a 6-bit ECC.
  - Reading the same address twice triggers two instances of flash read operations.
  - During a read operation, two additional wait states are included in the memory wait states to parse the data buffer.
  - Every instance of writing data triggers a write operation in the flash memory.
  - A 6-bit ECC complements each 16-bit data word.

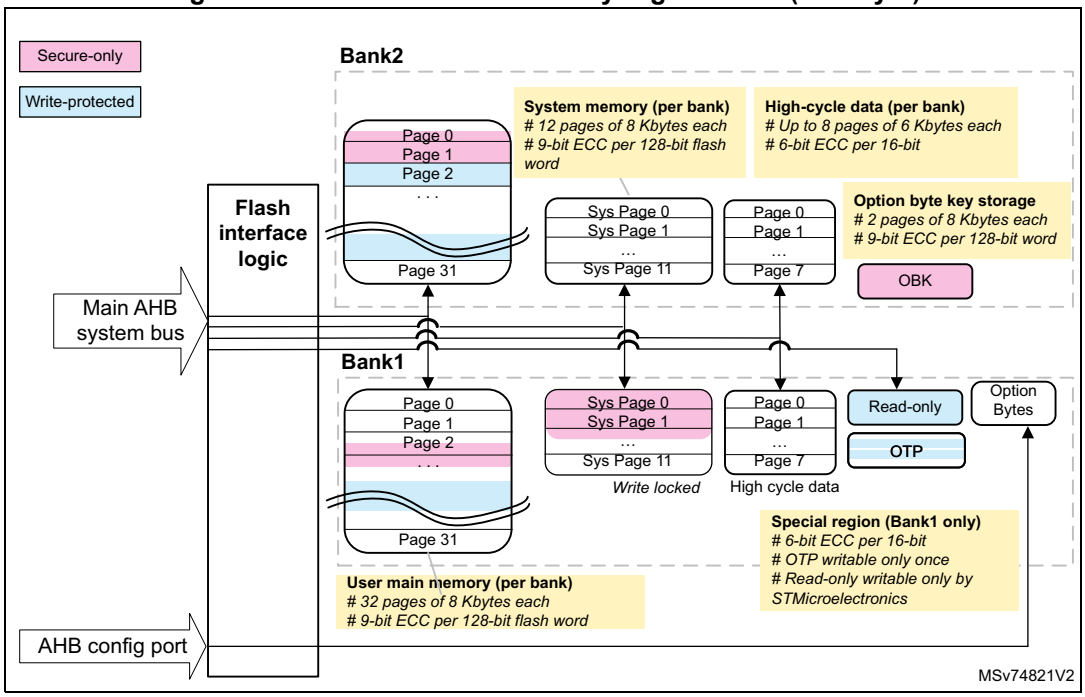
By default, the full AHB memory range is cacheable. For regions where caching is not practical (OTP, RO, data area), MPU must be used to disable local cacheability.

7.3.3 Flash memory architecture and usage

Flash memory architecture

Figure 17 shows the organization supported by the embedded flash memory.

Figure 17. Embedded flash memory organization (512kbyte)



The embedded nonvolatile flash memory is composed of:

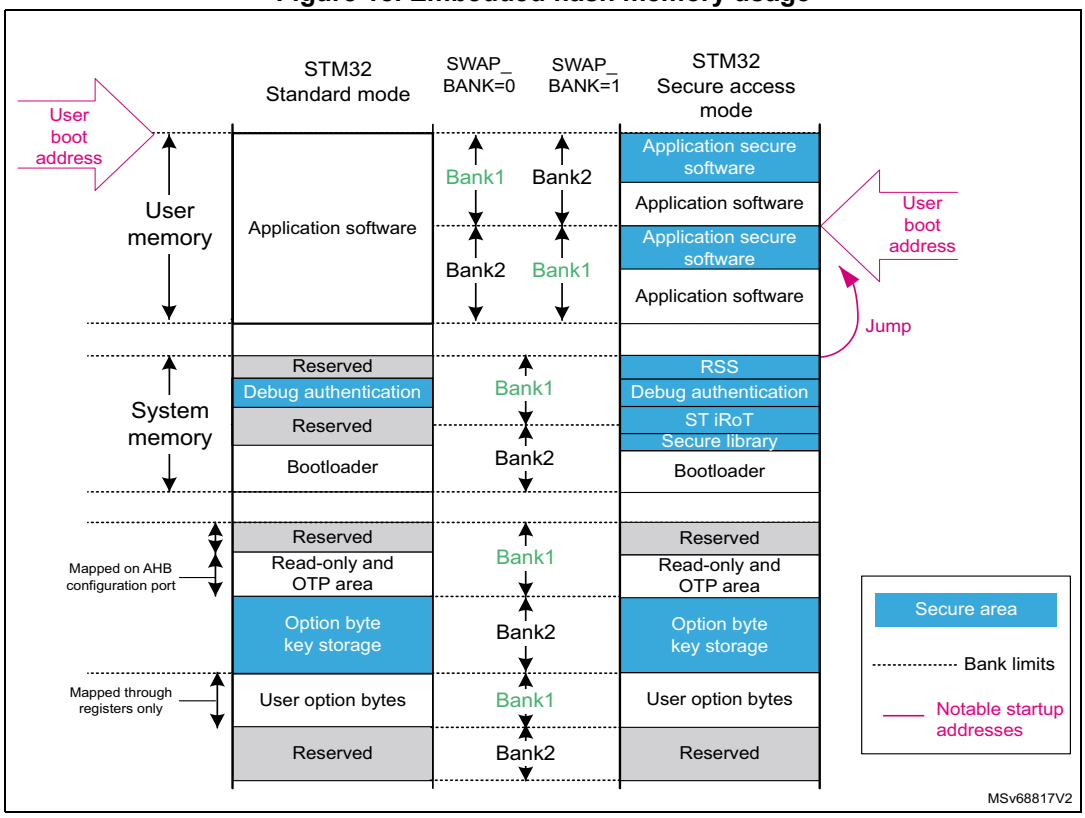
- A main memory block, organized in two equal banks. Each bank is divided in pages of 8 Kbytes, and features flash-word rows of 128 bits + 9 bits of ECC per word.
- A system memory block of 192 Kbytes, divided into two 96-Kbyte banks. Each bank is divided in 8-Kbyte pages. The system flash memory is ECC-protected (9-bit ECC per 128-bit word).
- A set of nonvolatile option bytes, loaded at reset by the embedded flash memory and accessible by the application software only through the AHB configuration register interface.
- A 2-Kbyte one-time-programmable (OTP) area that can be written only once by the application software.
- A 2-Kbyte read-only area. It contains a unique device ID and product information.
- Two memory pages (2 x 8 Kbytes) of secure key storage, OBKeys.
- Up to 16 pages of user flash memory with a high cycling capability (100 K cycles) for data, (four pages per bank for STM32G53x/54x and eight pages per bank for STM32G55x/56x devices).

The overall flash memory architecture and its corresponding access interface are summarized in *Table 72*.

Partition usage

Figure 18 shows how the embedded flash memory is used by STMicroelectronics and by the application software.

Figure 18. Embedded flash memory usage



The user and system memories are used differently, according to the product state and other option byte settings:

- The user memory contains the application code and data, while the system memory is used with root secure services (RSS), the debug authentication code, and the STM32 bootloader. When a reset occurs, the core jumps to the boot address configured through the BOOT pin, the BOOTADD/SBOOTADD option bytes, and the product state.
- The unique boot entry (BOOT_UBE) is set to either STiRoT, located in the system flash memory, or iRoT in the user flash memory. This feature is available only on cryptography-enabled devices.
- The secure service library in the system flash memory is available in secure mode.
- If the debugger is attached to the product, the entry point is the debug authentication policy, used to unlock the device via the SBS when attached to the debugger. A digital signature is required to perform a regression to the product state, where debug is allowed.
- If BOOTADD/SBOOTADD is not yet configured, dedicated libraries can be used for the secure boot. They are located in the system flash memory:
  - ST libraries in the system flash memory assist the application software boot with special features, such as the secure boot and secure firmware installation (SFI-RSS).

– STiRoT (ST immutable Root of trust) secure software in the user flash memory is used for the secure firmware update and provisioning (SFU).

Note: For more information on the option byte setup for boot, refer to Section 7.4.7.

7.3.4 FLASH read operations

Read operation overview

Read access to the main and system flash memories operates as follows:

- A 128-bit read data buffer is associated with each bank, storing the last data read. If several consecutive read access operations request data belonging to the same flash data word (128 bits), data are read directly from the current data read buffer, without triggering additional flash read operations. This mechanism occurs each time read access is granted. When read access is rejected for security reasons, the corresponding read error response is issued by the embedded flash memory and no read operation of the flash memory is triggered.
- The buffer is invalidated by write access, OBK swap, user option change request, read access to 16-bit flash word area or any erase operation.

Read access to the OTP, RO, and flash high-cycle data operates as follows:

1. A flash data word of 137 bits is read and stored in a temporary buffer.
2. The interface parses the 137-bit data word and selects the 16- or 32-bit data requested.
3. While parsing the 137-bit data word, two wait states are added, and the AHB bus is stalled.
4. If the application reads an OTP data or flash high-cycle data not previously written, **a double ECC error** is reported and **only a word full of set bits is returned** (see *Section 7.3.10* for details). The read data (in 16 bits) is stored in the FLASH_ECCDR register.
5. Reading the same address twice triggers two read operations in the flash memory.
6. For 8-bit access instances, an AHB bus error is generated.

Note: The embedded flash memory can perform single-error correction and double-error detection while read operations are in execution (see Section 7.3.9).

Instruction prefetch

The Cortex®-M33 processor fetches instructions and literal pools (constants/data) over the C-bus and through the instruction cache, if enabled. The prefetch block aims at increasing the efficiency of C-bus access operations when the instruction cache is enabled, by reducing the cache refill latency.

Prefetch is efficient in case of sequential code. Prefetch in the flash memory allows the next sequential instruction line to be read from the memory, while the current instruction line is being filled in the instruction cache and executed by the CPU.

Prefetch is enabled by setting the PRFTEN bit in the FLASH access control register (FLASH_ACR). The PRFTEN must be set only if at least one wait state is needed to access the flash memory.

Adjusting read timing constraints

The embedded clock must be enabled and running before reading data from a nonvolatile memory.

To read data from the memory correctly, the number of wait states (LATENCY) must be correctly programmed in the access control register (FLASH_ACR), according to the main AHB interface clock frequency and the internal voltage range of the device (V_{core}).

Table 39 shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded flash memory clock frequency, and the supply voltage range.

Table 39. Recommended number of wait states and programming delay

Number of wait states (LATENCY)	Programming delay (WRHIGHFREQ)	Interface clock frequency vs. V_{CORE} range(1)		
		VOS2 range 1.05 to 1.15 V	VOS1 range 1.15 to 1.26 V	VOS0 range 1.30 to 1.40 V
0 WS (1 FLASH clock cycle)	00	0 to 30 MHz	0 to 34 MHz	0 to 42 MHz
1 WS (2 FLASH clock cycles)		30 to 60 MHz	34 to 68 MHz	42 to 84 MHz
2 WS (3 FLASH clock cycles)	01	60 to 90 MHz	68 to 102 MHz	84 to 126 MHz
3 WS (4 FLASH clock cycles)		90 to 120 MHz	102 to 137 MHz	126 to 168 MHz
4 WS (5 FLASH clock cycles)	10	120 to 150 MHz	137 to 171 MHz	168 to 210 MHz
5 WS (6 FLASH clock cycles)		150 to 170 MHz	171 to 205 MHz	210 to 252 MHz
6 WS (7 FLASH clock cycles)	11	N/A	205 to 240 MHz	252 to 280 MHz

1. Voltage range from 1.26 to 1.30 V is not supported.

Adjusting system frequency

After power-on, the embedded flash memory is clocked by the 64 MHz high-speed internal oscillator (HSI), with a voltage range set at a scaled value of VOS2: a conservative two wait-state latency is specified in the FLASH_ACR register (see *Table 39*).

When changing the bus frequency, the application software must follow the sequence described below, to tune the number of wait states required to access the memory.

To increase the CPU frequency:

1. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in *Table 39.*
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACR register.
3. Modify the embedded flash memory clock source and/or the clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new embedded flash memory clock source and/or the new AHB clock prescaler value are taken into account by reading back the embedded flash memory clock source status and/or the prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).

To decrease the CPU frequency:

1. Modify the embedded flash memory clock source and/or the clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
2. Check that the embedded flash memory new clock source and/or the new clock prescaler value are taken into account by reading back the embedded flash memory clock source status and/or the AHB interface prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).
3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in *Table 39.*
4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACR register.

Error code correction (ECC)

The memory embeds an error correction mechanism. Single-error correction and double-error detection are performed for each read operation. For more details, refer to *Section 7.3.9*.

Read errors

When the ECC mechanism is unable to correct the read operation, the memory reports read errors, as described in *Section 7.9.10: Error correction code error (ECCC, ECCD)*.

Read interrupts

See *Section 7.10: FLASH interrupts* for details.

7.3.5 FLASH program operations

Program operation overview

Program operation consists of issuing write commands. The memory supports the execution of only one write command at a time. Write-while-write is not supported. Nothing prevents overwriting a non-virgin flash word, but it is not recommended as this may lead to invalid data and inconsistent ECC code.

User flash, OBK storage and system flash memories pages

For the user and system flash memories, a 9-bit ECC complements each 128-bit data flash word. In this case, the embedded flash memory must always perform write operations to nonvolatile memory with a 128-bit word granularity. Once the write buffer is full (128 bits), the Busy flag is set, and a programming operation is triggered.

There is a write buffer common to Bank1 and 2, which supports multiple write-access types (128, 64, 32, 16, or 8 bits). The application can decide to write from 8 bits to 128 words. In this case, a force-write mechanism to the 128 bits + ECC is used (refer to the FW bit of the FLASH_CR/FLASH_SCR register).

When the write request is issued to the memory, any new write request stalls the main AHB bus. Moreover, while a write operation is ongoing, any new read request for the same bank stalls the main AHB bus.

OTP, RO, flash high-cycle data

When the target memory is OTP, RO or flash high-cycle data pages, a 6-bit ECC code complements each 16-bit data flash word. The embedded flash memory supports 16- or 32-bit write operations (8-bit write operations are not supported).For 8-bit operations, write access requests are ignored. There is no write data buffer and each write access request triggers a write operation in the flash memory.

Note: The OTP area is typically write-protected on the final product, as described in Section 7.3.10.

The write protection check is performed at the reception of the write request (during the address phase).Write protection is not checked at the output of the write buffer anymore. If a write protection violation is detected, the write operation is canceled, and a write protection error (WRPERR) is raised in the FLASH_SR/FLASH_SSR register.

Note: For write protections of the main flash memory, ICP, and OTP see Section 7.6: FLASH security and protections.

Monitoring ongoing write operations

The application software can use a status flag located in FLASH_SR/FLASH_SSR to monitor ongoing write operations. Since only one operation is possible at a time, this flag indicates if any operation (write, erase, option change) is ongoing, whatever the bank.

- **BSY**: indicates that an effective write, erase, option byte change, OBK swap, OBK alt page erase, or PUF operation is ongoing in the nonvolatile memory. This flag is not dedicated to a specific bank. It is set when an operation is starting on the memory, whatever the bank. An operation is triggered by:
  - An erase (FLASH_CR/SCR.STRT)
  - A write (FLASH_CR/SCR.PG + AHB write)
  - An option modification (FLASH_OPTCR.OPTSTRT)
  - PUF_LAUNCH
  - OBKeys page swap and OBKeys page erase (FLASH_OBKCFGR/SOBKCFGR.SWAP_SECT_REQ and FLASH_OBKCFGR/SOBKCFGR.ALT_SECT_ERASE)

They are cleared when the current operation ends, or in case of an error.

- **WBNE**: this bit indicates that the embedded flash memory is waiting for new data to complete the 128-bit write buffer. In this state the write buffer is not empty. It is reset as soon as the application software fills the write buffer, forces the write operations by using the FW bit in FLASH_CR/SCR, or an error is detected. When WBNE is high, it is not possible to launch an erase operation, an option modification, an OBK swap, or an OBK alternate page erase operation on the flash memory.
- **DBNE**: this bit indicates that the data buffer for parsing 16-bit data is not empty:
  - 16-bit data write access is received, and the data buffer is being filled. It is set at the reception of a valid write access, and reset as soon as the write request preparation has been processed.

Note: If the memory is busy at the reception of the AHB write request, the CPU execution is stalled.

Enabling write operations

Before programming the user flash memory in Bank1 or in Bank2, the application software must ensure that the PG bit is set to 1 in FLASH_CR/SCR. If not, an unlock sequence must be used (see *Section 7.6.7: Flash memory configuration protection*), and the PG bit must be set.

When an option byte or an option byte key must be modified, or a mass erase must be started, the application software must ensure that FLASH_OPTCR is unlocked. If this is not the case, an unlock sequence must be used (see *Section 7.6.7: Flash memory configuration protection*).

A separate mechanism with similar use exists for FLASH_OBKCFGR/SOBKCFGR. The control register must be unlocked prior to the start of any OBKeys storage modification. Writing the correct sequence to the FLASH_OBKKEYR unlocks FLASH_OBKCFGR. FLASH_SOBKKEYR is linked to FLASH_SOBKCFGR, as described in the *Section 7.6.7: Flash memory configuration protection*.

Note: The application software must not unlock an already unlocked register, otherwise this register remains locked until the next system reset.

If needed, the application software can update the programming delay, as described in *Adjusting programming timing constraints*.

Writing to the FLASH control register FLASH_CR/SCR and FLASH_OPTCR

The FLASH_CR/SCR, FLASH_OPTCR and FLASH_OBKCFGR/SOBKCFGR registers are not accessible in write mode when the BSY bit is set. Any attempt to write these registers while the BSY bit is set causes the AHB bus to stall until the BSY bit is cleared.

Single-write sequence

The recommended single-write sequence is the following:

1. Make sure a protection mechanism does not prevent programming.
2. Check that no flash memory operation is ongoing by checking the BSY bit in the FLASH_SR/SSR register and CDBNE bits in the FLASH_SR/SSR register. Check that the write buffer is empty by checking the WBNE bit in the FLASH_SR/SSR register.
3. Check and clear all the error flags due to a previous programming/erase operation.
4. Unlock the FLASH_CR/SCR register, as described in *Section 7.6.7: Flash memory configuration protection* (only if this register is not already unlocked).
5. Enable write operations by setting the PG bit in the FLASH_CR/SCR register
6. Write one flash word at the aligned address.

Note: The WBNE flag indicates if the 128-bit write buffer is waiting for new data.

Note: No erase request, options change request, PUF, or OBK operation is allowed between the first write and the completion of the write operation.

7. Wait for the BSY bit to be cleared in the corresponding FLASH_SR/SSR register.
8. Clear the PG bit in theFLASH_CR/SCR register if there are no more programming requests.

If step 6 is executed incrementally (for example, byte per byte), the write buffer can become partially filled. In this case, the application software can decide to force-write what is stored in the write buffer by using the FW bit in the FLASH_CR/SCR register. In this particular

case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW bit has no effect.

Note: Using a force-write operation prevents the application from, at a later stage, updating the missing bits with a value different from 1. This can lead to unexpected or inconsistent data, or ECC.

Adjusting programming timing constraints

Program operation timing constraints depend on the memory clock frequency, which impacts performance directly. If timing constraints are too tight, the nonvolatile memory does not operate correctly. If they are too lax, the programming speed is not optimal.

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACR register. Refer to *Table 39: Recommended number of wait states and programming delay* for the recommended programming delay, depending on the memory clock frequency.

The FLASH_ACR configuration register is common to both banks.

The application software must check that no program/erase operation is ongoing before modifying WRHIGHFREQ.

Caution: Modifying WRHIGHFREQ while programming/erasing the memory can corrupt its content.

Programming errors

When a program operation fails, an error is reported, as described in *Section 7.9: FLASH error management*.

Programming interrupts

See *Section 7.10: FLASH interrupts* for details.

7.3.6 FLASH erase operations

Erase operation overview

The memory can perform erase operations on 8-Kbyte user pages, on one user flash memory bank, or on two user flash memory banks (for example mass erase). For more details on the user flash memory, ICP, user options, and OTP erase protection, see *Section 7.6: FLASH security and protections*.

Erase commands are issued through the AHB configuration interface. The memory supports one operation at a time, if it receives simultaneously a write and an erase request an error flag is raised, and both operations are canceled. See *Section 7.9: FLASH error management* for details.

After successful completion of erase, all the bytes in the erased flash pages are set to the flash default 0xFF value.

Erase and WRP

If the application software attempts to erase a write-protected user page, the page erase operation is aborted, and the WRPERR flag is raised in the FLASH_SR/SSR register, as described in *Section 7.9.2: Nonsecure write protection error (WRPERR)*.

Flash busy

The busy signal is described in *Section : Monitoring ongoing write operations*.

Writing to the FLASH control register FLASH_CR/SCR and FLASH_OPTCR

Refer to *Section : Writing to the FLASH control register FLASH_CR/SCR and FLASH_OPTCR*.

Enabling erase operations

Before erasing a page, the application software must make sure that FLASH_CR/SCR is unlocked. If this is not the case, an unlock sequence must be used (see *Section 7.6.7: Flash memory configuration protection*).

Note: The application software must not unlock a register that is already unlocked, otherwise this register remains locked until next system reset. This can be used to deliberately lock out a register from further access.

Similar constraints apply to bank erase requests.

Standard flash page erase sequence

To erase an 8- or 6-Kbyte data user page without security protections, proceed as follows:

1. Make sure a protection mechanism does not prevent page erase (WRP, secure flag, HDP).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SR register, and that the write buffer is empty by checking the WBNE bit in the FLASH_SR register.
3. Check and clear all nonsecure error flags due to previous programming/erase operations. Refer to *Section 7.9: FLASH error management* for details.
4. Unlock the FLASH_CR register, as described in *Section 7.6.7: Flash memory configuration protection* (only if this register is not already unlocked).
5. Set the BKSEL bit, the PER bit, and the PNB bitfield in the FLASH_CR register. BKSEL indicates in which physical bank the page must be erased, then PER indicates a page erase operation, while PNB contains the target page number. In case of a data page, use the number of the corresponding regular page. Refer to *Figure 19* to *Figure 24* for orientation.
6. Set the STRT bit in the FLASH_CR register.
7. Wait for the BSY bit to be cleared in the FLASH_SR register.
8. The STRT bit is automatically cleared at the end of the page erase, or in case of an error.
9. Clear PER in the FLASH_CR register if there are no more page erase requests to be issued.

Note: If another erase flag is requested simultaneously to the page erase, a PGSERR error is generated.

Secure flash memory page erase sequence

To erase an 8- or 6-Kbyte data secure configured user page, proceed as follows:

1. Make sure write protection or an HDP mechanism does not prevent page erase.
2. Ensure that no flash memory operation is ongoing (by checking the BSY and DBNE bits in the FLASH_SSR register), and that the write buffer is empty (by checking the WBNE bit in the FLASH_SSR register).
3. Check and clear all secure error flags due to previous programming/erase operations. Refer to *Section 7.9: FLASH error management* for details.
4. Unlock the FLASH_SCR register, as described in *Section 7.6.7: Flash memory configuration protection* (only if this register is not already unlocked).
5. Set the BKSEL bit, the PER bit, and the PNB bitfield in the FLASH_SCR register. BKSEL indicates in which physical bank the page must be erased, then PER indicates a page erase operation, while PNB contains the target secure page number. In case of a data page, use the number of the corresponding regular page. Refer to *Figure 19* to *Figure 24* for orientation.
6. Set the STRT bit in the FLASH_SCR register.
7. Wait for the BSY bit to be cleared in the FLASH_SSR register.
8. The STRT bit is automatically cleared at the end of the page erase or in case of an error.
9. Clear PER in the FLASH_SCR register if there are no more secure page erase requests to be issued.

Note: If another erase flag is requested simultaneously to the page erase, a PGSERR error is generated.

Standard flash memory bank erase sequence

To erase a bank where no page is configured as secure:

1. Make sure a protection mechanism does not prevent page erase.
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SR register and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all nonsecure error flags due to previous programming/erase operation. Refer to *Section 7.9: FLASH error management* for details.
4. Unlock the FLASH_CR register, as described in *Section 7.6.7: Flash memory configuration protection* (only if this register is not already unlocked).
5. Set the BKSEL bit and the BER bit in the FLASH_CR register to the targeted physical bank (swap setting is ignored).
6. Set the STRT bit in the FLASH_CR register to start the bank erase operation. Then wait until the BSY bit is cleared in the FLASH_SR register.
7. The STRT bit is automatically cleared at the end of the erase sequence or in case of an error.
8. Clear BER in the FLASH_CR register if there is no other bank erase request to be issued.

Secure flash memory bank erase sequence

To erase a bank where all pages are configured with a secure flag:

1. Make sure a protection mechanism does not prevent page erase (for mass erase HDP is also considered, it cannot be fully executed from HDPL=2,3 if HDP is defined).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SSR register, and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the secure error flags due to previous programming/erase operation. Refer to *Section 7.9: FLASH error management* for details.
4. Unlock the FLASH_SCR register, as described in *Section 7.6.7: Flash memory configuration protection* (only if this register is not already unlocked).
5. Set the BKSEL bit and the BER bit in the FLASH_SCR register to the targeted physical bank (swap setting is ignored).
6. Set the STRT bit in the FLASH_SCR register to start the bank erase operation. Then wait until the BSY bit is cleared in the FLASH_SSR register.
7. STRT bit is automatically cleared at the end of erase sequence or in case of an error.
8. Clear BER in the FLASH_SCR register if there is no other bank erase request to be issued.

Flash mass erase sequence

To erase all pages of both banks, using nonsecure access, all pages must be configured as nonsecure. The application software can set the MER bit to 1 in the FLASH_CR register, as described below:

1. Make sure no protection mechanisms prevent mass erase (*Section 7.6.7: Flash memory configuration protection*).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SR register.
3. Check and clear all nonsecure error flags due to previous programming/erase operations. Refer to *Section 7.9: FLASH error management* for details.
4. Unlock the FLASH_CR register as described in *Section 7.6.7: Flash memory configuration protection* (only if the registers are not already unlocked).
5. Set the MER bit to 1 in FLASH_CR register.
6. Set the STRT bit in the FLASH_CR register. Then wait until the BSY bit is cleared in the FLASH_SR register.
7. The STRT bit is cleared automatically at the end of the erase sequence, or in case of an error.
8. Clear MER in the FLASH_CR register.

Secure flash memory mass erase sequence

To erase all pages of both banks simultaneously, using secure access, all pages must be configured as secure. The application software can set the MER bit to 1 in the FLASH_SCR register, as described below:

1. Make sure no protection mechanisms prevent mass erase (*Section 7.6.7: Flash memory configuration protection*).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SSR register, and that the write buffer is empty by checking the WBNE bit in the FLASH_SSR register.
3. Check and clear all secure error flags due to previous programming/erase operations. Refer to *Section 7.9: FLASH error management* for details.
4. Unlock the FLASH_SCR register as described in *Section 7.6.7: Flash memory configuration protection* (only if the registers are not already unlocked).
5. Set the MER bit to 1 in FLASH_SCR register.
6. Set the STRT bit in the FLASH_SCR register. Then wait until the BSY bit is cleared in the FLASH_SSR register.
7. The STRT bit is cleared automatically at the end of the erase sequence, or in case of an error.
8. Clear MER in the FLASH_SCR register.

Note: Mass and bank erase also erase high-cycle data pages aliased from the erased bank.

7.3.7 PUF operations

PUF serves as HUK for secure storage using SAES. The PUF value is based on hardware technology that requires a special initialization procedure.

PUF initialization is performed by PUF_Launch and subsequent cryptographic test. The test is needed to establish that the PUF retrieved from the HW remains the same across the device operating envelope (voltage, temperature, for example).

During the product manufacturing a PUF_REF_Value is stored in the flash read-only area (refer to *Section 61: Device electronic signature*).

This value is the result of a cryptographic operation as the PUF value is never accessible. To verify the PUF, the application must reproduce the exact same cryptographic operation and retrieve the same value. This verification is not needed in PRODUCT_STATE = Open, where actual PUF value is also not used.

Verification process consists of following steps:

1. PUF_Launch: Initialize the HW IP (if PRODUCT_STATE = Open, jump to end)
2. In nonsecure code, with HDPL1 (SBS_NEXTHDPLCR), EPOCH forced to zero (SBS_EPOCHSELCR), ... SAES engine set with ECB mode, 256-bit DHUK selected use SAES to encrypt ClearPufRef value.
3. Compare the result with PUF_REF_Value stored in the read-only area.
4. If different, return to step 1.

The process is implemented in the Cube package under SAES example folder.

The plaintext input value for SAES is the following:

uint8_t ClearPufRef[32] = {

0xD1U, 0xCEU, 0xD1U, 0xCEU, 0xD1U, 0xCEU, 0xD1U, 0xCEU,

0xD1U, 0xCEU, 0xD1U, 0xCEU, 0xD1U, 0xCEU, 0xD1U, 0xCEU,

0xCEU, 0xD1U, 0xCEU, 0xD1U, 0xCEU, 0xD1U, 0xCEU, 0xD1U,

0xCEU, 0xD1U, 0xCEU, 0xD1U, 0xCEU, 0xD1U, 0xCEU, 0xD1U,

};

Not following the procedure may cause problems using SAES based secure storage because it leads to inconsistency in cryptographic key value.

Note: PUF operations last for several milliseconds and use flash memory controller write services. These are not usable until the operation finishes.

PUF verification with TrustZone disabled

To launch a PUF verification, proceed as follows:

1. Check that no flash memory operation is ongoing by checking the NSBSY and NSDBNE bits in the FLASH_SR register, and that the write buffer is empty by checking the NSWBNE bit in the FLASH_SR register.
2. Check and clear all nonsecure error flags due to previous programming/erase operations. Refer to *Section 7.9: FLASH error management* for details.
3. Unlock the FLASH_CR register, as described in *Section 7.6.7: Flash memory configuration protection* (only if this register is not already unlocked).
4. Write NS_PUF_LAUNCH in FLASH_CR.
5. Wait for the NSBSY bit to be cleared in the FLASH_SR register.
6. The NS_PUF_LAUNCH bit is automatically reset by the hardware at the end of the operation. If NS_PUF_READY is set in FLASH_SR, the PUF is usable until the next power-off/standby.

PUF verification with TrustZone enabled

To launch a PUF verification, proceed as follows:

1. Check that no flash memory operation is ongoing by checking the SECBSY and SECDBNE bits in the FLASH_SSR register, and that the write buffer is empty by checking the SECWBNE bit in the FLASH_SSR register.
2. Check and clear all nonsecure error flags due to previous programming/erase operations. Refer to *Section 7.9: FLASH error management* for details.
3. Unlock the FLASH_SCR register, as described in *Section 7.6.7: Flash memory configuration protection* (only if this register is not already unlocked).
4. Write SEC_PUF_LAUNCH in FLASH_SCR.
5. Wait for the SECBSY bit to be cleared in the FLASH_SSR register.
6. The SEC_PUF_LAUNCH bit is automatically reset by hardware at the end of the operation.
7. If SEC_PUF_READY is set in FLASH_SSR, the PUF is usable until the next power-off/standby.

7.3.8 FLASH parallel operations

The flash memory is divided in two independent banks. The embedded flash memory interface supports a read operation in one bank, while an erase or write (RWW: read-while-write) operation is executed simultaneously in the other bank. It does not support write-while-write, nor read-while-read. The same is valid for the high-cycle data area, system flash libraries, option bytes (located on Bank1), and OBK (located on Bank2).

In all cases, the sequences described in *Section 7.3.4*, *Section 7.3.5* and *Section 7.3.6* apply.

7.3.9 FLASH error protections

Error correction codes (ECC)

The embedded flash memory supports an error correction code (ECC) mechanism, based on the SECDED algorithm, to correct single errors and detect double errors.

This mechanism uses nine ECC bits per 128-bit flash word, and applies to the user and system memories. For read-only, OTP, and flash high-cycle data, six ECC bits per 16-bit word are used. A double ECC error is generated for an OTP or flash high-cycle data virgin word (for example, an OTP word with all 22 bits at 1). When this OTP or flash high-cycle data word is no longer virgin, the ECC error disappears.

More specifically, during each read operation from a 128-bit flash word, the embedded flash memory retrieves the 9-bit ECC information, computes the ECC of the flash word, and compares the result with the reference value. If these do not match, the corresponding ECC error is raised, as described in *Section 7.9.10: Error correction code error (ECCC, ECCD)*.

During each programming operation, a 9-bit ECC code complements each 128-bit data flash word, and the resulting 137-bit flash word information is written in the nonvolatile memory.

A similar mechanism applies to read-only and OTP areas, but with 6-bit ECC for 16-bit data.

7.3.10 OTP and RO memory access

The OTP and RO memories are accessed through the main AHB interface. The OTP memory is accessible at addresses 0x08FF F000 to 0x08FF F7FF, and the read-only section is accessible from 0x08FF F800 to 0x08FF FFFF.

FLASH one-time programmable area

The embedded flash memory offers a 2048-byte memory area dedicated to application non-confidential, one-time-programmable (OTP) data. This area is composed of 1024 words of 16 bits (plus 6 bits of ECC). It cannot be erased and can be written only once. The OTP area can be accessed through the main AHB interface from the address range 0x08FF F000 to 0x08FF F7FE.

OTP data can be programmed by the application software in 16-bit chunks. Overwriting an already programmed 16-bit half word is not supported.

Note: The OTP area is empty when the device is delivered by STMicroelectronics.

When reading OTP data with a single error corrected or a double error detected, the embedded flash memory reports read errors, as described in *Section 7.9.10: Error correction code error (ECCC, ECCD)*.

OTP write protection

OTP data are organized as 32 blocks of 32 OTP words, as shown in *Table 40*. An entire OTP block can be protected (locked) from write access by setting the LOCKBLi bit (i = 0 to 31) corresponding to each OTP block in the FLASH_OTPBLR register. A block can be write-protected, whether it has been programmed (even partially) or not.

The OTP block locking operation is irreversible, regardless of the product life state.

Note: The OTP area can be accessed only in read mode.

Table 40. Flash memory OTP organization

OTP block	AHB address	AHB word		Lock bit
		[31:16]	[15:0]	
Block 0	0x08FF F000	OTP001	OTP000	LOCKBL0
	0x08FF F004	OTP003	OTP002	
	...			
	0x08FF F03C	OTP031	OTP030	
Block 1	0x08FF F040	OTP033	OTP032	LOCKBL1
	0x08FF F044	OTP035	OTP034	
	...			
	0x08FF F07C	OTP063	OTP062	
Block 2	0x08FF F080	OTP065	OTP064	LOCKBL2
	0x08FF F084	OTP067	OTP066	
	...			
	0x08FF F0BC	OTP95	OTP94	
...				
Block 31	0x08FF F7C0	OTP993	OTP992	LOCKBL31
	0x08FF F7C4	OTP995	OTP994	
	...			
	0x08FF F7FC	OTP1023	OTP1022	

OTP write sequence

Follow the sequence below to write an OTP word:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the FLASH_SR register and that the data buffer is empty by checking the DBNE bit in the FLASH_SR register.
2. Check and clear all error flags due to previous programming/erase operations.
3. Set the PG bit in the FLASH_CR register.
4. Check the protection status of the target OTP word (see *Table 40*). The corresponding LOCKBLi bit must not be set to 1.
5. Write two OTP words (32 bits) corresponding to the 4-byte aligned address shown in *Table 40*. Alternatively, the application software can program separately the 16-bit MSB or 16-bit LSB. In this case, the first 16-bit write operation starts immediately without waiting for the second one.
6. Wait for the BSY bit to be cleared in the FLASH_SR register.
7. Clear the PG bit in FLASH_CR register if there are no further programming requests in the bank.
8. Optionally, lock the OTP block using LOCKBLi to prevent further data changes.

Note: Do not write a 16-bit OTP word twice, as it generates an error.

Writing OTP data at byte level is not supported, and generates a bus error.

To avoid data corruption, it is important to complete the OTP write process (for example, by reading back the OTP value), before starting an option change.

Flash read-only area

The embedded flash memory offers a 2-Kbyte area to store read-only data. This area can be accessed through the AHB main port, and is protected by a robust ECC scheme, as detailed in *Section 7.3.9*.

The read-only information (programmed by STMicroelectronics) that can be used by the application software is detailed in *Table 41*.

Table 41. Read-only public data organization

Read-only data name	Address	Comment
Unique device ID	0x08FF F800	U_ID[31:0]
	0x08FF F804	U_ID[63:32]
	0x08FF F808	U_ID[96:64]
Flash memory size/ package	0x08FF F80C	Flash memory size[15:0]/Package code[15:0]
Trim data	0x08FF F810 to 0x08FF F81B	Sensor and voltage reference trimming values
PUF_REF_Value	0x08FF F81C to 0x08FF F83B	Reference value used for PUF initialization
Trim data	0x08FF F83C - 0x08FF F84B	USBPD trim data
Reserved	0x08FF F84C to 0x08FF FFFF	Reserved information, except for values described elsewhere in user documentation

7.3.11 Flash high-cycle data

The embedded flash memory offers up to 96 Kbytes (maximum) memory area with high-cycling capability (100 kcycles) to store data and emulate EEPROM (up to 48 Kbytes for STM32G53x/54x and up to 96 Kbytes per bank for STM32G55x/56x devices). It can be accessed through the AHB system port from address 0x0900_0000 to 0x0901_7FFF (see *Figure 19*). It is mapped in the 8 (or 4) last pages of Bank1 and 2. This area is protected by a robust
6-bit ECC, which allows a read and write granularity of 16 bits at the expense of having the page size shrunk to 6 Kbytes.

A threshold per bank (EDATA(1/2)_STRT) is programmable to determine the beginning of the data flash area. By default, the entire memory is used for code.

For example, if 48 Kbytes of data are needed in Bank1, in FLASH_EDATA1R set EN to 1, and STRT to 7. If no data are needed in Bank2, reset FLASH_EDATA2R.EN to 0. In this case the data are accessible from address 0x0900_0000 to address 0x0900_BFFF for Bank1.

For greater efficiency, it is always recommended to use pages on the other bank for flash high-cycle data, so that the application benefits from the RWW capability of the dual-bank arrangement.

When banks are swapped (FB_MODE setting in SBS): the flash high-cycle data in Bank 2 are accessible from 0x0900_000 to 0x0900_BFFF, and the data in Bank 1 are accessible from 0x0900_C000 to 0x0901_7FFF.

A bus error is generated on:

- An attempt to access an address between 0x0900_0000 to 0x0901_7FFF and this address is not valid, not enabled, or STRT incorrect).
- An attempt to fetch instructions from the pages occupied by the flash high-cycle data area.

Erasing the data area page is possible by normal erase request for corresponding user flash page (24-31).

Protections and security of high-cycle area are detailed in *Section 7.6.9: Flash high-cycle data protections*.

Figure 19. Flash high-cycle data memory map on STM32G55xB/56xB devices (128 Kbytes)

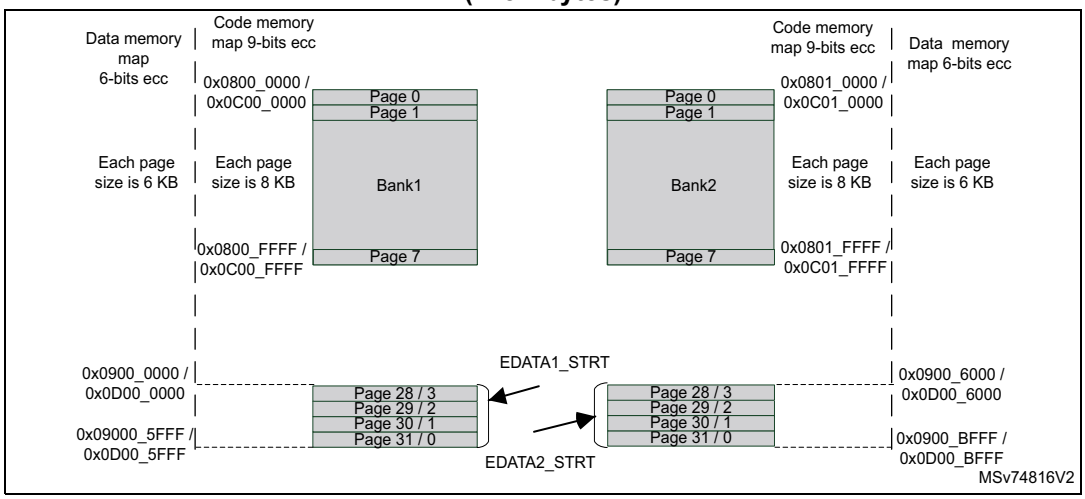


Figure 20. Flash high-cycle data memory map on STM32G55xC/56xC devices (256 Kbytes)

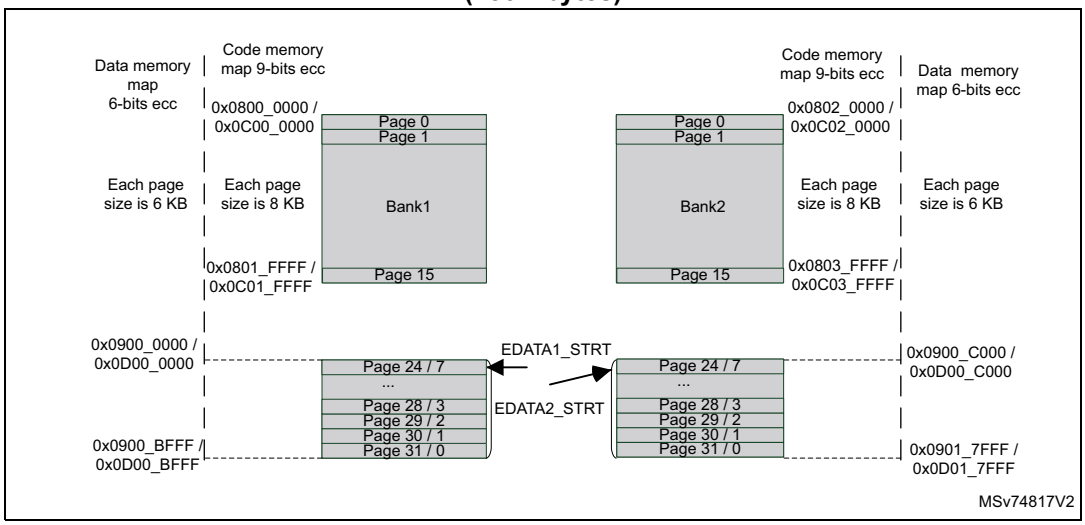
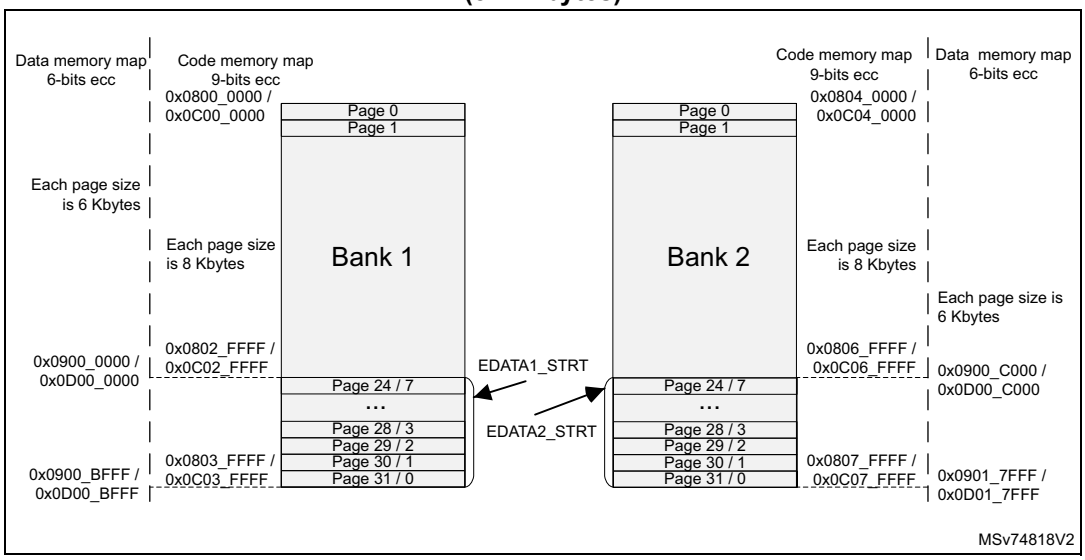


Figure 21. Flash high-cycle data memory map on STM32G55xE/56xE devices (512 Kbytes)



Note: When the flash high-cycle data area on Bank1 is enabled, the code memory map is not continuous to the following flash segment using 512 Kbytes of user flash memory on STM32G55xE/56xE.

Figure 22. Flash high-cycle data memory map on STM32G53x8/54x8 devices (64 Kbytes)

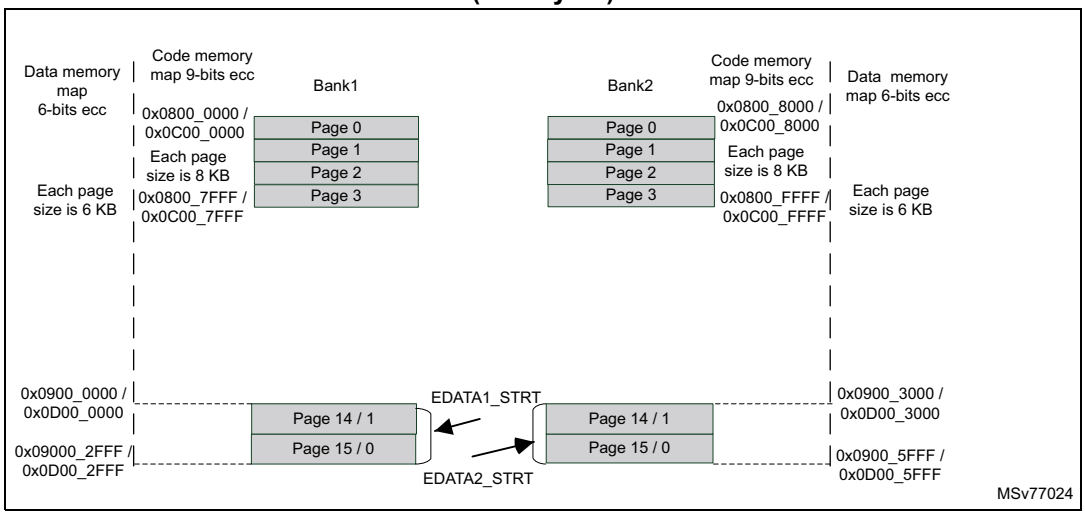


Figure 23. Flash high-cycle data memory map on STM32G53xB/54xB devices (128 Kbytes)

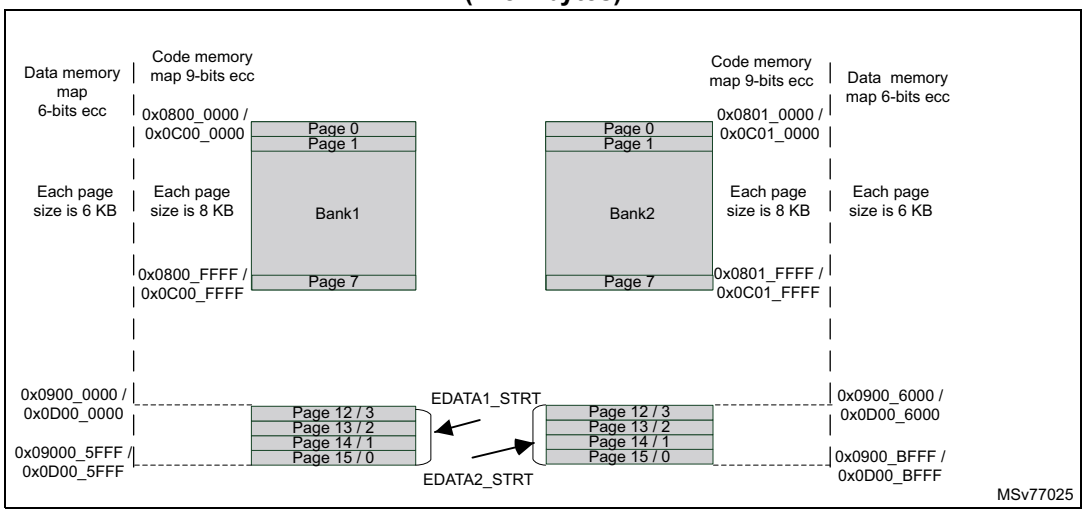
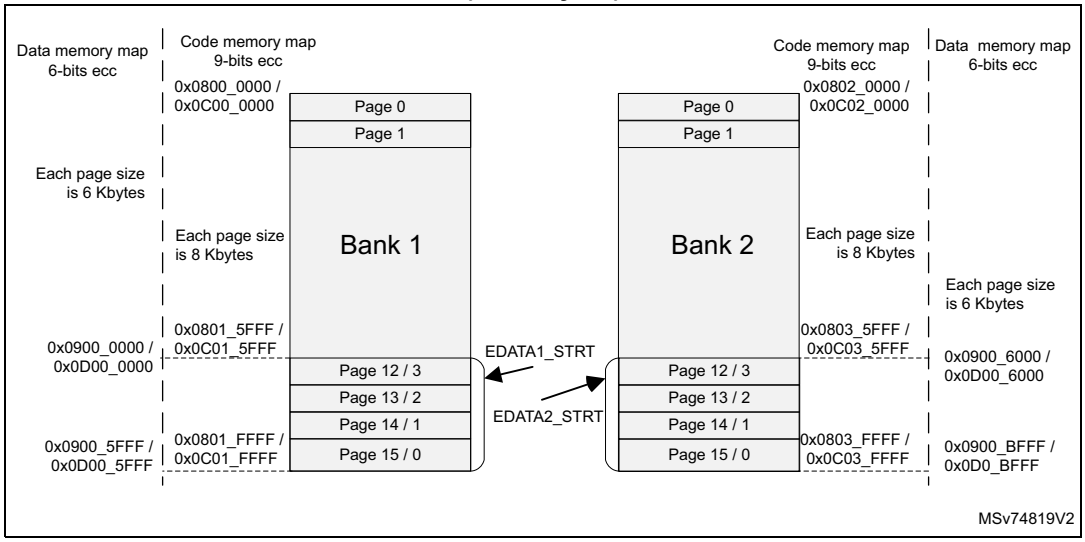


Figure 24. Flash high-cycle data memory map on STM32G53xC/54xC devices (256 Kbytes)



Note: When flash high-cycle data area on Bank1 is enabled, the code memory map is not continuous on STM32G53x/54x.

7.3.12 Flash bank swapping

Bank1 and Bank2 can be swapped for the user flash memory. This feature can be used for a firmware upgrade to switch the device to the new firmware. Bank swapping uses a user option byte flag controlled by the SWAP_BANK byte of the FLASH_OPTSR3 register and mapping controls available in the SBS. As explained in *Section 14.3.9: SBS flash banks control*, the SBS value of FB_MODE is also the definitive indicator of the currently used bank mapping.

Bank-specific settings for data area and security attributes follow the original bank and its contents. Control bit BKSEL always refers to the physical bank, not the mapping indicated by the FB_MODE setting.
The FB_MODE setting is loaded from the SWAP_BANK option only after a system reset or POR.

Table 42. Memory map and swapping options (64 Kbytes)

<table>
<tr><th rowspan="2">Area</th><th colspan="2">Corresponding bank</th><th rowspan="2">Start address</th><th rowspan="2">End address</th><th rowspan="2">Size</th><th rowspan="2">Region name</th></tr>
<tr><th>FB_MODE=0xC3</th><th>FB_MODE=0xB4</th></tr>
<tr><td rowspan="12">User main memory</td><td rowspan="4">Bank1</td><td rowspan="4">Bank2</td><td>0x0800 0000</td><td>0x0800 1FFF</td><td>8K</td><td>Page 0</td></tr>
<tr><td>0x0800 2000</td><td>0x0800 3FFF</td><td>8K</td><td>Page 1</td></tr>
<tr><td>0x0800 4000</td><td>0x0800 5FFF</td><td>8K</td><td>Page 2</td></tr>
<tr><td>0x0800 6000</td><td>0x0800 7FFF</td><td>8K</td><td>Page 3</td></tr>
<tr><td rowspan="4">Bank2</td><td rowspan="4">Bank1</td><td>0x0800 8000</td><td>0x0800 9FFF</td><td>8K</td><td>Page 0</td></tr>
<tr><td>0x0800 A000</td><td>0x0800 BFFF</td><td>8K</td><td>Page 1</td></tr>
<tr><td>0x0800 C000</td><td>0x0800 DFFF</td><td>…</td><td>Page 2</td></tr>
<tr><td>0x0800 E000</td><td>0x0800 FFFF</td><td>8K</td><td>Page 3</td></tr>
<tr><td rowspan="12">System memory</td><td colspan="2" rowspan="5">Bank1</td><td>0x0BF8 0000</td><td>0x0BF8 1FFF</td><td>8K</td><td>System 1 Page 0</td></tr>
<tr><td>0x0BF8 2000</td><td>0x0BF8 3FFF</td><td>8K</td><td>System 1 Page 1</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>0x0BF9 6000</td><td>0x0BF9 7FFF</td><td>8K</td><td>System 1 Page 11</td></tr>
<tr><td rowspan="7">0x0BF9 8000</td><td rowspan="7">0x0BF9 9FFF</td><td rowspan="7">8K</td><td rowspan="7">System 2 Page 0</td></tr>
<tr><td colspan="2" rowspan="6">Bank2</td></tr>
<tr><td>0x0BF9 A000</td></tr>
<tr><td>0x0BF9 BFFF</td></tr>
<tr><td>8K</td></tr>
<tr><td>System 2 Page 1</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>0x0BFA E000</td><td>0x0BFA FFFF</td><td>8K</td><td>System 2 Page 11</td></tr>
</table>

Table 43. Memory map and swapping options (128 Kbytes)

Area	Corresponding bank		Start address	End address	Size	Region name
	FB_MODE=0xC3	FB_MODE=0xB4				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8K	Page 0
			0x0800 2000	0x0800 3FFF	8K	Page 1
			…	…	…	…
			0x0800 E000	0x0800 FFFF	8K	Page 7
	Bank2	Bank1	0x0801 0000	0x0801 1FFF	8K	Page 0
			0x0801 2000	0x0801 3FFF	8K	Page 1
			…	…	…	…
			0x0801 E000	0x0801 FFFF	8K	Page 7
System memory	Bank1		0x0BF8 0000	0x0BF8 1FFF	8K	System 1 Page 0
			0x0BF8 2000	0x0BF8 3FFF	8K	System 1 Page 1
			…	…	…	…
			0x0BF9 6000	0x0BF9 7FFF	8K	System 1 Page 11
	Bank2		0x0BF9 8000	0x0BF9 9FFF	8K	System 2 Page 0
			0x0BF9 A000	0x0BF9 BFFF	8K	System 2 Page 1
			…	…	…	…
			0x0BFA E000	0x0BFA FFFF	8K	System 2 Page 11

Table 44. Memory map and swapping options (256 Kbytes)

Area	Corresponding bank		Start address	End address	Size	Region name
	FB_MODE=0xC3	FB_MODE=0xB4				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8K	Page 0
			0x0800 2000	0x0800 3FFF	8K	Page 1
			…	…	…	…
			0x0801 E000	0x0801 FFFF	8K	Page 15
	Bank2	Bank1	0x0802 0000	0x0802 1FFF	8K	Page 0
			0x0802 2000	0x0802 3FFF	8K	Page 1
			…	…	…	…
			0x0803 E000	0x0803 FFFF	8K	Page 15

Table 44. Memory map and swapping options (256 Kbytes)

System memory	Bank1	0x0BF8 0000	0x0BF8 1FFF	8K	System 1 Page 0
		0x0BF8 2000	0x0BF8 3FFF	8K	System 1 Page 1
		…	…	…	…
		0x0BF9 6000	0x0BF9 7FFF	8K	System 1 Page 11
	Bank2	0x0BF9 8000	0x0BF9 9FFF	8K	System 2 Page 0
		0x0BF9 A000	0x0BF9 BFFF	8K	System 2 Page 1
		…	…	…	…
		0x0BFA E000	0x0BFA FFFF	8K	System 2 Page 11

Table 45. Memory map and swapping options (512 Kbytes)

Area	Corresponding bank		Start address	End address	Size	Region name
	FB_MODE=0xC3	FB_MODE=0xB4				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8K	Page 0
			0x0800 2000	0x0800 3FFF	8K	Page 1
			…	…	…	…
			0x0803 E000	0x0803 FFFF	8K	Page 31
	Bank2	Bank1	0x0804 0000	0x0804 1FFF	8K	Page 0
			0x0804 2000	0x0804 3FFF	8K	Page 1
			…	…	…	…
			0x0807 E000	0x0807 FFFF	8K	Page 31
System memory	Bank1		0x0BF8 0000	0x0BF8 1FFF	8K	System 1 Page 0
			0x0BF8 2000	0x0BF8 3FFF	8K	System 1 Page 1
			…	…	…	…
			0x0BF9 6000	0x0BF9 7FFF	8K	System 1 Page 11
	Bank2		0x0BF9 8000	0x0BF9 9FFF	8K	System 2 Page 0
			0x0BF9 A000	0x0BF9 BFFF	8K	System 2 Page 1
			…	…	…	…
			0x0BFA E000	0x0BFA FFFF	8K	System 2 Page 11

Bank swap with intermediate reset

To change the SWAP_BANK setting, follow the sequence below:

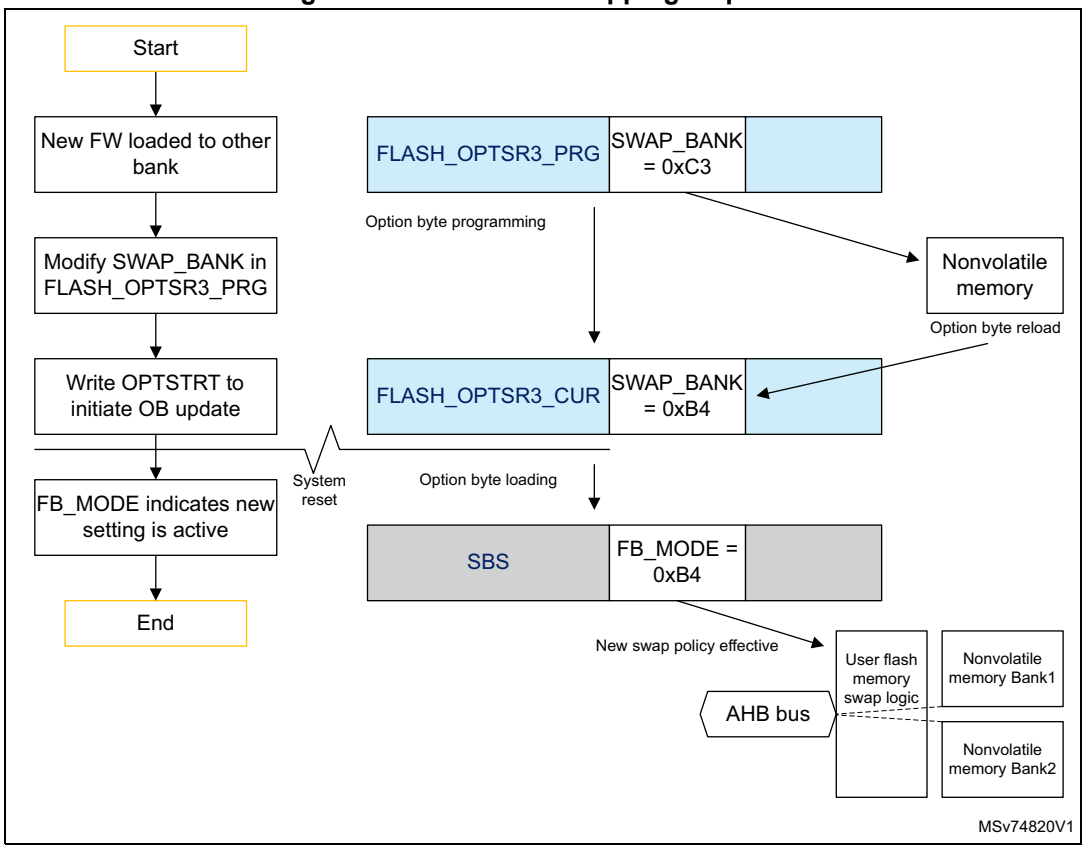
1. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SR/SSR register, and that the write buffer is empty by checking the WBNE bit in the FLASH_SR/SSR register.
2. Clear all error flags due to any previous operations.
3. Unlock the OPTLOCK bit, if not already unlocked.
4. Set the new desired SWAP_BANK value in the FLASH_OPTSR3_PRG register.
5. Start the option byte change sequence by setting the OPTSTRT bit in the FLASH_OPTCR register.
6. Once the option byte change has been completed, FLASH_OPTSR3_CUR contains the expected SWAP_BANK value, but FB_MODE has not yet been modified and the bank swapping is not yet effective.
7. Force a system reset or a POR. When the reset rises up, the bank swapping is effective and the new firmware shall be executed.

Note: The SWAP_BANK bit in FLASH_OPTSR3_CUR is read-only, and cannot be modified by the application software.

The SWAP_BANK option bit in FLASH_OPTSR3_PRG can be modified during any product state, provided the SWAP_LOCK protection value, controlled by the SBS, is 0xC3 (not engaged).

Figure 25 gives an overview of the bank swapping sequence.

Figure 25. Flash bank swapping sequence



On-the-fly bank swap

When not locked, the FB_MODE setting can be changed at any moment during runtime, resulting in an immediate remapping of the flash memory banks. This mechanism is useful for the implementation of on-the-fly field-upgrades with minimum down time. In case of security concerns, this bit can be locked out from modification (see details in the SBS chapter).

When the FB_MODE value is changed during runtime, the CPU fetches the next instruction from the other bank. To prevent a crash, essentially both banks need to contain the exact same code in the code segment performing the swap, interrupts should be disabled, no other flash access must be ongoing, and the new code needs to be able to successfully unroll the stack created by the old code.

This exact same code performing the swap is called unmodifiable code. The execution may continue to execute the updated code after the successful swap.

The recommended sequence of unmodifiable code is:

1. Disable interrupts
2. Disable and flush cache
3. Modify the interrupt vector table offset (if needed)
4. Swap the bank
5. Restore cache configuration
6. Enable interrupts

It's important to know that in the case of a reset, the boot bank is selected according to the SWAP_BANK setting. It's recommended to modify the setting to follow the latest functional code.

7.3.13 FLASH reset and clocks

Reset management

The embedded flash memory can be reset by a core domain reset, driven by the reset and clock control (RCC). The main effects of this reset are the following:

- All registers, except for option byte registers, are cleared, including read and write latencies. If the bank swapping option is changed, it is applied.
- Most control registers are automatically protected against write operations. To disable the protection, new unlock sequences must be used as described in *Section 7.6.7: Flash memory configuration protection*.

The memory can be reset by a power-on core domain reset, driven by the reset and clock control (RCC). When the reset falls, all option byte registers are reset. When the reset rises up, the option bytes are loaded, potentially applying new features. During this loading sequence, the device remains under reset, and the memory is not accessible.

The reset signal can have a critical impact on the memory: the content is not guaranteed if a device reset occurs during a write or erase operation.

Reset occurring during flash operation

If a reset occurs during a flash operation (programing, erase, or option change), the content of the memory is not guaranteed. It is mandatory for integrity to restart the operation. The status register FLASH_OPSR gives information about operations interrupted by a reset.

FLASH_OPSR.CODE_OP gives the opcode of the operation. *Table 46* indicates how to use FLASH_OPSR, and which operation is required.

Table 46. Recommended reactions to FLASH_OPSR contents

CODE_OP	Operation interrupted	OTP_OP	SYSF_OP	BK_OP	DATA_OP	Flash area	ADDR_OP min	ADDR_OP max	Recommended action
0x000	No operation ongoing while reset	0	0	0	0	-[1]	-	-	No extra action
0x001	Write operation	0	0	0/1	0	User flash	0x0000	0xFFFF	Erase page and rewrite
		0	1	0/1	0	System flash	0x0000	0x0FFF	
		1	0	0	0	OTP	0x0600	0x07FF	
		1	0	1	0	OBKeys[2]	0x0000	0x03FF	
		0	0	0/1	1	Data area[3]	0xF000	0xFFFF	
0x010	OBK alternate page erase	0	0	1	0	OBKeys	0	0	Relaunch the alternate page erase
0x011	page erase[4]	0	0	0/1	0	User flash	0x0000	0xFFFF	Relaunch page erase
		0	1	0/1	0	System flash	0x0000	0x0FFF	
0x100	Bank erase	0	0	0/1	0	User flash	-	-	Relaunch bank erase
0x101	Mass erase	0	0	0	0		-	-	Relaunch mass erase
0x110	Option change	0	0	0	0	User configuration	-	-	New attempt on option change
0x111	OBK swap page	0	0	1	0	OBKeys	-	-	Erase alternate page, reprogram new OBKeys and relaunch swap

1. Dash represents “does not matter”.
2. Depends on current OBK page.
3. Depends on EDATA setting in the OB.
4. Addresses indicated are aligned to page start, data area pages are erased by erase request to corresponding user flash memory page.

Clock management

The memory uses the microcontroller system clock (sys_ck), here the AHB interface clock.

7.4 FLASH option bytes

7.4.1 About option bytes

The memory includes a set of nonvolatile option bytes. They are loaded at power-on reset and can be read and modified only through configuration registers. This section documents:

- When option bytes are loaded.
- How application software can modify them.
- Their detailed list, together with their initial values (before the first option byte change, user default configuration).

7.4.2 Option byte loading

There are multiple ways of loading the option bytes:

- **Power-on wakeup**

  When the device is powered on, the embedded flash memory automatically loads all the option bytes. During the option byte loading sequence, the device remains under reset and the embedded flash memory cannot be accessed.

- **Wakeup from system standby**

  When the core power domain, which contains the embedded flash memory, is switched from Standby mode to Run mode, the embedded flash memory behaves as during a power-on sequence.

Note: During loading time the device is not under reset, unlike during a power-on sequence.

- **Dedicated option byte reloading by the application**

  When the user application successfully modifies the option byte content through the memory registers, the nonvolatile option bytes are programmed and the memory automatically reloads all option bytes to update the option registers.

Note: The option-byte read sequence is protected by error correction code. In case of error, the option bytes are loaded with default values (see Section 7.4.3: Option-byte modification), different from the initial values (user default configuration), and more restrictive.

7.4.3 Option-byte modification

Changing user option bytes

A user option-byte change operation can be used to modify the configuration and the protection settings saved in the nonvolatile option byte area of memory Bank1. It's important to account for a stall in execution from Bank1, when the option bytes are being modified.

Several rules are enforced when attempting to change a user option byte. These are summarized in *Section 7.4.8*. Failing to stick to those rules usually results in errors, as described in *Section 7.9.12*.

The embedded flash memory features two sets of option byte registers:

- The first register set contains the current values of the option bytes. Their names have the _CUR extension. All these registers are read-only. Their values are automatically loaded from the nonvolatile memory after power-on reset, wakeup from system standby or after an option byte change operation.

- The second register set allows the modification of the option bytes. Their names contain the _PRG extension. All “_PRG” registers can be accessed in read/write mode.

When the OPTLOCK bit in the FLASH_OPTCR register is set, modifying the FLASH_XXX_PRG registers is not possible.

When the OPTSTRT bit is set to 1, the memory checks the programming sequence (PGSERR) and the conditions described in *Section 7.4.8* (OPTCHANGEERR). If no error has been detected (PGSERR/OPTCHANGEERR), the flash memory launches the option byte modification and updates the option byte registers with _CUR extension.

If one of the condition described in *Section 7.4.8*, *Section 7.9.12* or *Section 7.9.5*, alternatively *Section 7.9.4* is not respected, the memory aborts the option byte change operation. In this case, the FLASH_XXX_PRG registers are not overwritten by the current option value. The user application can check what was wrong in their configuration.

Unlocking the option-byte modification

After reset, the OPTLOCK bit is set to 1 and the FLASH_OPTCR is locked. As a result, the application software must unlock the option configuration register before attempting to change the option bytes. The FLASH_OPTCR unlock sequence is described in *Section 7.6.7: Flash memory configuration protection*.

Option-byte modification sequence

To modify user option bytes, follow the sequence below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the FLASH_SR/SSR register, and that the write buffer is empty by checking the WBNE bit in the FLASH_SR/SSR register.
2. Check that the data buffer is empty (DBNE = 0) in the FLASH_SR/SSR register.
3. Clear all error flags due to a previous operation.
4. Unlock the FLASH_OPTCR register as described in *Section 7.6.7: Flash memory configuration protection*, unless the register is already unlocked.
5. Write the desired new option byte values in the corresponding option registers (FLASH_XXX_PRG).
6. Set the option byte start change OPTSTRT bit to 1 in the FLASH_OPTCR register.
7. Wait until BSY bit is cleared in the FLASH_SR/SSR register.
8. The OPTSTRT bit is cleared automatically at the end of the sequence (or in case of an error).
9. Reset the device. This step is always recommended, and even mandatory when one of the following option bytes is impacted:
   a) BOOTADDR
   b) SBOOTADD
   c) TZEN
   d) BOOT_UBE

Note: If a reset or power-down occurs while the option byte modification is ongoing, the original option byte value is kept. A new option byte modification sequence is required to program it.

7.4.4 Option-byte overview

Table 47 lists all user option bytes managed through the memory registers, as well as initial values before the first option byte change (production values configuration).

Table 47. Option-byte organization

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_OPTSR	Reserved	Reserved	BOOT_UBE								IWDG_STDBY	IWDG_STOP	Reserved	Reserved	Res.	Res.	PRODUCT_STATE								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Reserved	Reserved	
	0	0	0xB4								1	1	0	0	0	0	0xED								1	1	0	1	1	0	0	0
FLASH_OPTSR2	TZEN								Res.								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM3_ECC	SRAM2_ECC	SRAM1_ECC	SRAM3_RST	SRAM2_RST	SRAM1_RST	Res.	Res.
	0xC3								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0
FLASH_OPTSR3	Res.								SWAP_LOCKOUT								SWAP_BANK								Res.							
	0	0	0	0	0	0	0	0	0xC3								0xC3								0x00							
FLASH_SBOOTR	SBOOTADD[23:8]																SBOOTADD[7:0]								SBOOT_LOCK							
	0x0C00																0x00								0xC3							
FLASH_BOOTR	BOOTADD[23:8]																BOOTADD[7:0]								BOOT_LOCK							
	0x0800																0x00								0xC3							
FLASH_SECWM1R	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1END					Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1STRT				
	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_SECWM2R	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2END					Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2STRT				
	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_WRP1R	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPG1[31:28]	WRPG1[27:24]	...				WRPG1[7:4]	WRPG1[3:0]
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FLASH_WRP2R	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPG2[31:28]	WRPG2[27:24]	...				WRPG2[7:4]	WRPG2[3:0]
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 47. Option-byte organization (continued)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_ OTPBLR	LOCKBL																															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_ EDATA1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA1STRT		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_ EDATA2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA2STRT		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_ EPOCHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH																							
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FLASH_ SEPOCHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEPOCH																							
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FLASH_ HDP1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1END					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1STRT				
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FLASH_ HDP2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2END					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2STRT				
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
BL_COM_CFG	BL_COM_CFG[31:0]																															
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

7.4.5 Description of user and system option bytes

The general-purpose option bytes that can be used by the application are listed below:

- Watchdog management
  - IWDG_STOP: independent watchdog IWDG counter active in Stop mode if 1 (stop counting or freeze if 0)
  - IWDG_STDBY: independent watchdog IWDG counter active in Standby mode if 1 (stop counting or freeze if 0)
  - IWDG_SW: hardware (0) or software (1) IWDG watchdog control selection

Note: If the hardware watchdog "control selection" feature is enabled (set to 0), the watchdog is automatically enabled at power-on, thus generating a reset unless the watchdog key register is written into or the down-counter is reloaded before the end-of-count is reached.

Depending on the configuration of IWDG_STOP and IWDG_STDBY options, the IWDG can continue counting (1) or not (0) when the device is, respectively, in Stop or Standby mode. When the IWDG is kept running during Stop or Standby mode, it can wake up the device from these modes.

- Reset management
  - NRST_STDBY generates a reset when entering Standby mode if cleared to 0.
  - NRST_STOP generates a reset when entering Stop mode if cleared to 0.

Note: Whenever a Standby (Stop) mode entry sequence is successfully executed, the device is reset instead of entering Standby (Stop) mode if NRST_STDBY (NRST_STOP) is cleared to 0.

- Bootloader configuration
  - After OBL, if the STMicroelectronics bootloader is activated, it reads the contents of BL_COM_CFG and only attempts to establish communication on selected interfaces. This setting allows using the bootloader safely in applications where it would otherwise cause trouble by reconfiguring some GPIOs.

At delivery, the values programmed in the user option bytes are the following:

- Watchdog management
  - IWDG (also known as WDGLS_CD) active in Standby and Stop modes: 0x1
  - IWDG (also known as WDGLS_CD) not automatically enabled at power-on: 0x1
- All bootloader interfaces enabled

Refer to *Section 7.11: FLASH registers* for details.

7.4.6 Description of data protection option bytes

The option bytes that can be used to enhance data protection are listed below:

- PRODUCT_STATE[7:0]: a product life cycle state (see *Section 7.6.11: Product state transitions* for details).
- WRPGn1/2: write protection option of the corresponding group of four consecutive pages in Bank1 (respectively Bank2). It is active low. Refer to *Section 7.6.8: Write protection* for details.
  - Bit N: Group embedding pages 4 x N to 4 x N + 3
  - Number of bits configurable depends on the total user flash size
- SECWMx: TrustZone® secure-only watermark area definition (refer to *Section 7.6.1: TrustZone® security protection* for details).
  - STRT (respectively END) contains the first (respectively last) page of the secure access only zone in Bankx.
  - The number of configurable bits depends on the total user flash memory size.
- TZEN: this nonvolatile option can be used to deactivate the secure access mode, as described in *Section 7.6: FLASH security and protections*. For security reasons, the TZEN is stored with redundancy on eight bits. While the TZEN OB value is set

immediately after programming, to actually disable the isolation, a reset is required to set the TZ_STATE in SBS.

When TZEN is changed, the secure watermark settings and secure boot address may be reset to default to prevent a deadlock in case the previous values were incorrect.

- HDPx: secure hide protection - HDPL exclusive area control in the user flash memory.
  - The number of configurable bits depends on the total user flash memory size.

The factory-programmed values of the data protection option bytes are the following:

- Flash bank erase operations do not impact watermarked secure data areas.
- Secure watermark area protections are disabled (the start addresses is higher than the end addresses).
- Write protection is disabled (all option byte bits set to 1).
- TrustZone® secure access mode is enabled (TZEN option byte value = 0xB4).

Refer to *Section 7.11: FLASH registers* for details.

7.4.7 Description of boot address option bytes

The option bytes that can be used to configure the appropriate boot address for an application are listed below:

- PRODUCT_STATE
- BOOT_UBE: available only on cryptography-enabled devices.
  Selects either STiRoT or the user flash memory as the default boot address. Also used by RSS(SFI) to choose either the bootloader or STiRoT as the next stage.
- BOOTADD: selects the default boot address when TZ is disabled.
- SBOOTADD: selects the default boot address when TZ is enabled.
- BOOT_LOCK/SBOOT_LOCK: protects the boot configuration from further modification attempts.
- SWAP_LOCKOUT: reset value for the SWAP_LOCK setting in the SBS. If engaged, bank swapping is disabled.
- SWAP_BANK: bank swapping option, set to 1 to swap user flash banks after boot. If SWAP_LOCK is active in the SBS, the value in SWAP_BANK is locked out from modification until reset.

When STMicroelectronics delivers the device, the PRODUCT_STATE is Open, BOOT_LOCK/SBOOT_LOCK is not set (0xC3) and the BOOT_UBE is set to the user flash memory (0xB4). Addresses are SBOOTADD = 0x0C00 0000, BOOTADD = 0x0800 0000.

Refer to *Section 7.11: FLASH registers* for details.

7.4.8 Specific rules for modifying option bytes

On top of OPTLOCK bit and register access rules, other protective means for selected security-sensitive option byte fields are in place.

Multiple option bytes can be modified simultaneously, but if they rely on each other for protection, both states are checked. For example, when trying to modify PRODUCT_STATE and TZEN simultaneously, the resulting state must be coherent with the rules. With a few exceptions, listed in this section, failing to uphold the rules results in raising the OPTCHANGEERR flag (*Section 7.9.12* for additional details).

Table 48. Specific OB modifying rules

Option byte	HDPL	TZ secure	Value	Product state
PRODUCT_STATE	1(1)	Some values only when TZ is enabled	Set of possible transitions	Set of possible transitions (*Table 39*)
HDP	1(2)	(3)	-	Refer to *Table 49*
TZEN	-	-	-	Regression, Open, or Provisioning
EPOCH (secure)	0	-	PRG > CUR	Regression
EPOCH	-	-	PRG > CUR	NS-Regression, Regression
SECWM	-	Secure mode only	-	Refer to *Table 49*
BOOT_UBE	-	-	-	Open or Provisioning, SBOOT_LOCK
SBOOTADD	-	Secure mode only(2)	-	
BOOTADD	-	-	-	Open or Provisioning, BOOT_LOCK disabled
BL_COM_CFG	-	-	-	Open or Provisioning, both BOOT_LOCK/SBOOT_LOCK disabled
LOCKBL	-	-	One way switch	-
SWAP_LOCKOUT	-	-	-	Open or Provisioning
SWAP_BANK	-	-	Fixed if SWAP_LOCK is set	-
SBOOT_LOCK	-	Secure mode only	-	Open, Regression, or Provisioning to unlock
BOOT_LOCK	-	-	-	

1. Most transitions are possible regardless of HDPL, only a few require specific HDPL, see text below.
2. No OPTCHANGEERR raised in violation of this.
3. Dash means there is no limitation from this side.

Even in the closed PRODUCT_STATE progression, some OBs can still be modified, if all the other constraints are satisfied. An overview is presented in *Table 49*.

Table 49. OB modifiable in closed product

PRODUCT_STATE	Modifiable OBs
TZ-Closed	SWAP_BANK, LOCKBL, PRODUCT_STATE, SECWM, HDP
Closed	SWAP_BANK, LOCKBL, PRODUCT_STATE
Locked	SWAP_BANK, LOCKBL

Specific rules must be respected to update the following OB:

- **PRODUCT_STATE**

  PRODUCT_STATE transitions follow a state machine with two types of transition. Locking down the product is allowed without restriction. Opening the product is possible only when using a debug interface and following a digital signature

verification. The regression to a less secure state results in the erasure of the protected content. More details are given in *Section 7.6.11: Product state transitions*.

Summary: Selected changes are only possible in HDPL1. Some states are accessible only when TZEN is enabled.

- **HDP**

  Can only be modified in HDPL1. Must not be set to overlap with the flash high-cycle data area.

- **TrustZone® access mode (TZEN)**

  Can only be changed in Open, Regression, and Provisioning.

- **SEPOCH and EPOCH**

  The value programmed must be greater than current value.The increment is done in specific product states: Regression and NS-Regression.

- **Secure watermark area (SECWM1/2 STRT and END)**

  Can be changed only in secure mode (TZ_state = 0xB4). Automatically reset to default value when TZEN is enabled. Must not be set to overlap with flash high-cycle data area.

- **BOOT_UBE**

  Available only on cryptography enabled devices.
  Can only be changed in product states open for debug, like Open, and also in Provisioning. BOOT_LOCK must be disabled to change.

- **SBOOTADD**

  Can only be changed in Open and Provisioning. Locked by SBOOT_LOCK. Automatically reset to default value when TZEN is enabled.

- **BOOTADD**

  Can only be changed in Open and Provisioning. Locked by BOOT_LOCK.

- **LOCKBL**

  Can only be changed freely in one direction. A permanent irreversible switch.

- **SWAP_BANK**

  Not modifiable when SWAP_LOCK is 0xB4 (set) in the SBS.

- **SWAP_LOCKOUT**

  Can only be changed in Open and Provisioning. If engaged, modification of SWAP_BANK and FB_MODE is locked out (not possible) and the controls remain in default state.

- **SBOOT_LOCK**

  Can only be changed freely in locking direction. Unlock is possible in Open, Provisioning, and Regression.

- **BOOT_LOCK**

  Can only be changed freely in locking direction. Unlock is possible in Open, Provisioning, and Regression.

Note: For all user option bytes above, the default values are loaded and Tamper is signaled when a double ECC error occurs during OBL.

7.5 Option-byte key management

This section describes the key storage tied with the SAES and OBK-HDPL value set by the SBS (refer to *Section 14.3.8: SBS hardware secure storage control*). The OBKs are stored in 128 bits and protected by a 9-bit ECC (SECDED). They are written alternatively in two pages.

7.5.1 OBK loading

OBKs are memory-mapped and accessible through the main AHB bus (C-Bus), starting from the address 0x0FFD 0000. A 9-bit ECC complements each 128-bit data flash word.

Note: Read access is allowed in the current and alternate page, except for the last address of the current and alternate page. In case of a read access, no error is reported but the read data is always 0x0 (see Section 7.6.12: OBK protection).

7.5.2 OBK access per HDPL level

Table 50 describes the option-byte key areas. The key storage is not dedicated to a particular key, and the usage is defined by the application.

An option-byte key can be accessed only if the OBK-HDPL (set in SBS) matches the HDPL associated with the storage offset, as indicated below (refer to *Section 14.3.8: SBS hardware secure storage control*). If this is not the case, an OBKERR error is raised.

Table 50. Option-byte key area

OBK-HDPL	Address offset start	Address offset end	Comment
-	0x0000	0x00FF	Reserved
1	0x0100	0x08FF	OEM iRoT keys
2	0x0900	0x0FFF	uRoT, OS, or secure application
3	0x1000	0x17FF	HDPL 3 secure keys
3	0x1800	0x1FEF	HDPL 3 nonsecure keys

7.5.3 OBK programming sequence

The sequence to write OBK is the same as a write operation to the main flash memory, except that the “alternate page” (ALT_SECT) bit must be set to write in the alternate OBK page. The ALT_SECT bit is checked each time a write access request is received. If it does not match the one stored in the write buffer, an error is raised.

Note: Before writing in the alternate OBK page, the user must ensure that the entire key space (128, 256, or 512 bits) is free by reading the alternate OBK page.

If the alternate page is not free, erase the page using the ALT_SECT_ERASE command.

Secure programming method

Using the secure register if TrustZone® is active (TZ_STATE = 0xB4):

1. Program the ALT_SECT bit.
2. Make sure protection mechanism does not prevent programming (TZ, PRIV and OBK-HDPL).
3. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SSR register.
4. Check and clear all secure error flags due to previous operation.
5. Unlock the FLASH_SCR register only if this register is not already unlocked, as described in *Section 7.6.7: Flash memory configuration protection*.
6. Enable write operations by setting the SECPG bit in the FLASH_SCR register.
7. Write one flash-word corresponding to 16-byte data starting at a 16-byte aligned address.

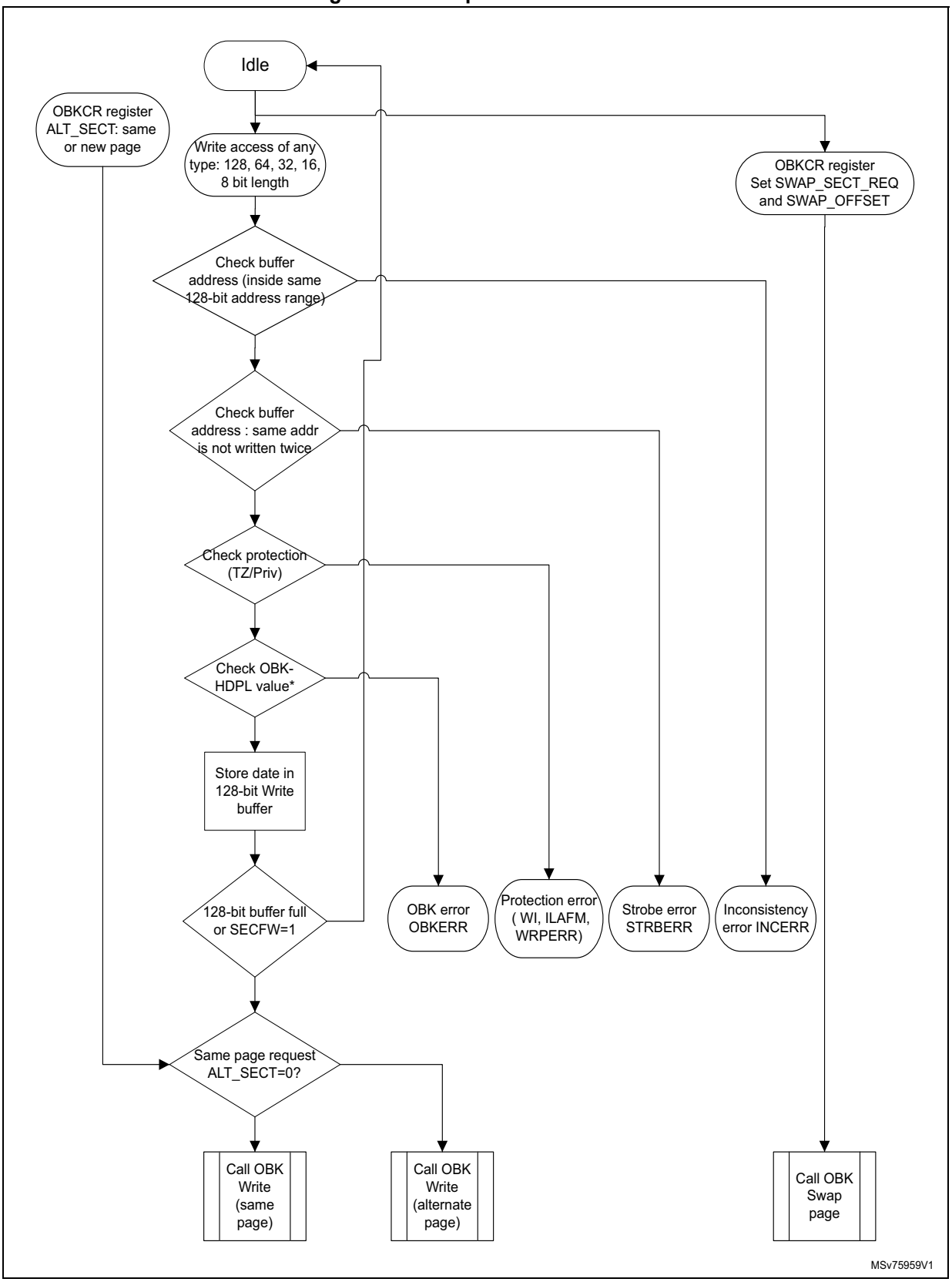
Note: The WBNE flag indicates if the 128-bit write buffer is waiting for new data.

Note: No erase request, options change request, OBK swap page, or OBK alternate page erase is allowed between the first write and the completion of a flash write operation.

8. Wait for the BSY bit to be cleared in the corresponding FLASH_SSR register.
9. Clear the PG bit in the FLASH_SCR register if there is no further programming requests in the bank.

If step 7 is executed incrementally (byte per byte), the write buffer can become partially filled. In this case, the application software can decide to force-write what is stored in the write buffer by using the FW bit in the FLASH_SCR register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW bit has no effect.

Figure 26. OBK protection checks

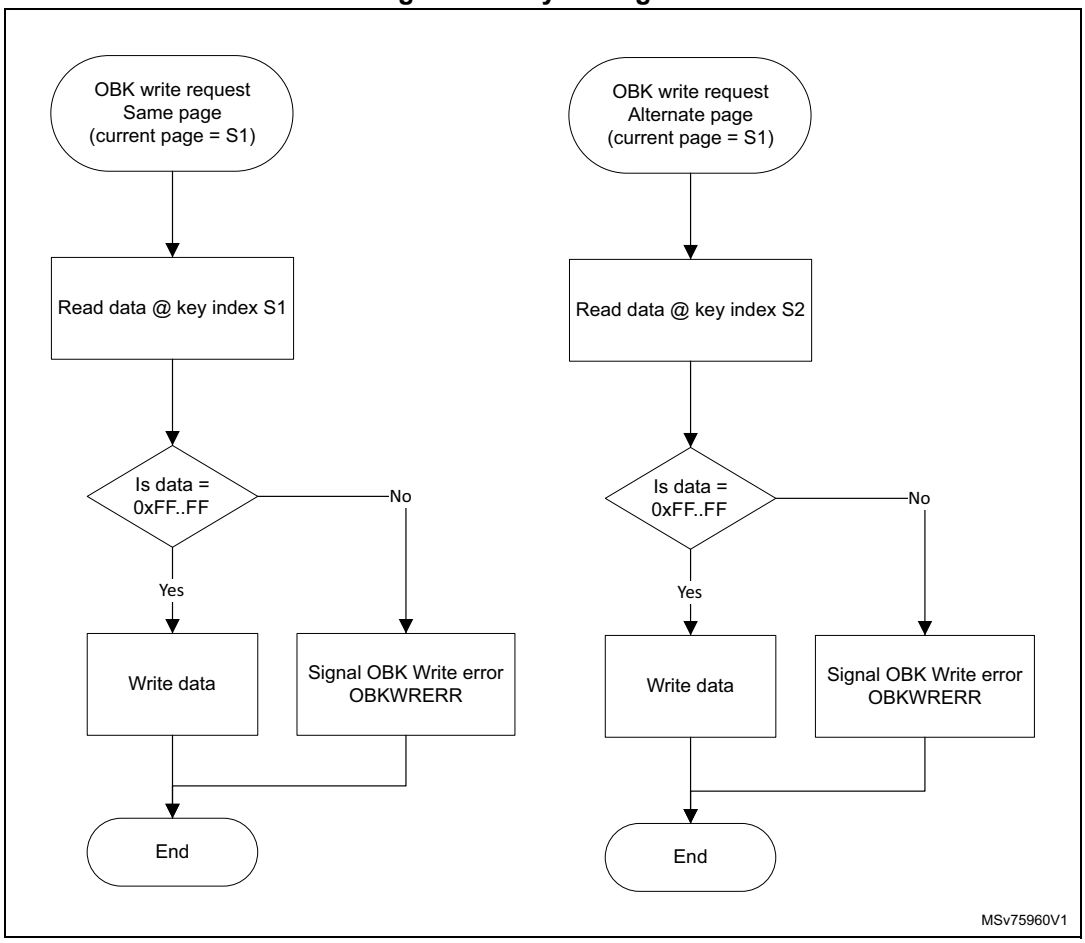


7.5.4 OBK programming finite state machine

When the OBK write buffer is filled, the OBK programing is launched automatically either in the current OBK page or in the alternate OBK page, depending on the ALT_SECT bit.

If the address is not virgin, the OBKWRERR is raised.

Figure 27. Key writing flow



7.5.5 OBK swap page

The user can request a swap of the option byte keys by setting the bit SWAP_SECT_REQ in the FLASH_OBKCFGR/SOBKCFGR register. The number of keys swapped are defined by the SWAP_OFFSET value in the FLASH_OBKCFGR/SOBKCFGR register.

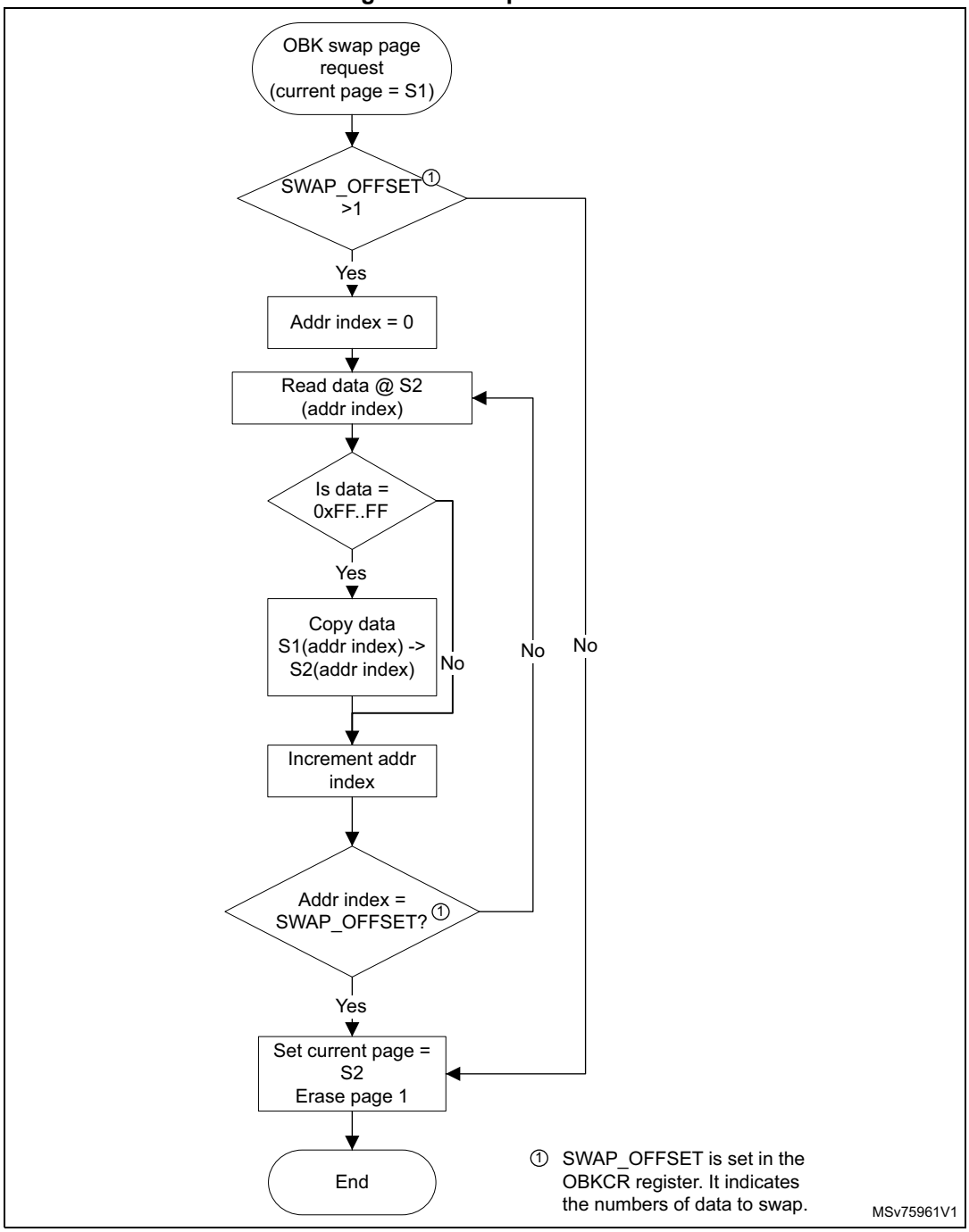
The swap is limited by the OBK-HDPL value and only the following SWAP_OFFSET values are permitted:

- OBK_HDPL1_OFFSET = 144
- OBK_HDPL2_OFFSET = 256
- OBK_HDPL3SEC_OFFSET = 384
- OBK_HDPL3NS_OFFSET = 511

The programming sequence is the following:

1. Check that no flash memory operation is ongoing by checking the BSY and DBNE bit in the FLASH_S/NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_S/NSSR register.
2. Check and clear all secure/nonsecure error flags due to previous operations.
3. Unlock the OBKCFGR/SOBKCFGR register if not unlocked already by using FLASH_OBKKEYR/SOBKKEYR.
4. Define the numbers of keys to be swapped with the SWAP_OFFSET field in FLASH_OBKCFGR/SOBKCFGR. SWAP_OFFSET must be equal to or greater than OBK_HDPL<N-1>_OFFSET.
5. Set the SWAP_SECT_REQ bit in the FLASH_OBKCFGR/SOBKCFGR register.
6. Wait for the BSY bit to be cleared. In case of an error, the OBKERR flag is raised.
7. SWAP_SECT_REQ is cleared automatically at the end of the operation or in case of an error.

Figure 28. Swap workflow



While the OBK swap operation is ongoing, the BSY flag is set and no other operation (write, erase, user option change) can be launched simultaneously.

7.5.6 OBK alternate page erase

The user can request the erasure of the alternate OBK page by setting the ALT_SECT_ERASE bit in the FLASH_OBKCFGR/SOBKCFGR register.

The programming sequence must be:

1. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SR/SSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SR/SSR register.
2. Check and clear all secure/nonsecure error flags due to previous operations.
3. Set the ALT_SECT_ERASE bit in the FLASH_OBKCFGR/SOBKCFGR register.
4. Wait for the BSY bit to be cleared.
5. ALT_SECT_ERASE is cleared automatically at the end of the operation or in case of an error.

7.6 FLASH security and protections

To protect sensitive information from unwanted operations (such as reading confidential areas, illegal programming of immutable pages, or malicious flash memory erasing), the FLASH implements the following protection mechanisms:

- TrustZone®-backed watermark and block security protection
- Temporal isolation protection (HDP)
- Configuration protection
- User flash write protection
- Device nonvolatile security life cycle and application boot state management
- OTP locking

The microcontroller can have TrustZone® active (TZ_STATE = 0xB4) or inactive (TZ_STATE = 0xC3).

When TrustZone® is enabled, the flash memory can be accessed in four basic modes:

- nonsecure and unprivileged
- nonsecure and privileged
- secure and privileged
- secure and unprivileged

When TrustZone® is disabled, only two modes are possible:

- nonsecure and unprivileged
- nonsecure and privileged

The flash memory interface evaluates the access restrictions in the following order:

1. TrustZone® security
2. Write protection (for write access)
3. HDP
4. Privilege

7.6.1 TrustZone® security protection

The global TrustZone® system security is activated by default, using the setting TZEN in the FLASH_OPTSR2_PRG register. However, the actual state of TZ is determined by the SBS and the setting of TZ_STATE (refer to *Section 14.3.6: SBS boot control*).

When TrustZone® is active (TZ_STATE = 0xB4), the following additional security features are available:

- Secure watermark-based user options bytes defining secure areas.
- Secure or nonsecure block-based areas can be configured on the fly after reset. This is a volatile secure area.
- Additional product states associated (TZ-Closed, NS-Regression) provide protection of the secure domain against external access (debug or bootloader).
- Erase or program operations can be performed in secure or nonsecure mode with an associated configuration bit. When TrustZone® is disabled (TZ_STATE = 0xC3), the features above are deactivated and all secure registers are RAZ/WI.
- The nonsecure access address space starts at 0x0800 0000 and the secure address space for the same physical memory starts at 0x1800 0000. The mapping is identical, but the address determines whether the TZ isolation allows or denies the memory access.

Active TrustZone® security

When TrustZone® is activated (TZ_STATE = 0xB4), the secure BOOTADD and the secure watermark areas are set to a secure default value. Default settings are also used in case of a corrupted configuration. In this default state, the user flash memory is secure as shown in the table below.

Table 51. Default secure watermark

Secure watermark in case of configuration error	Security attribute
STRT = 0 END = 0x1F	All flash memory is secure

Illegal access generation

Nonsecure access to a secure flash memory area is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the FLASHIE illegal access interrupt is enabled in the TZIC_IER2 register.

Nonsecure access to a secure flash register generates an illegal access event. An illegal access interrupt is generated if the FLASH_REGIE illegal access interrupt is enabled in the TZIC_IER2 register.

Table 52. Flash memory TZ protection summary

TZ protection (TZ_STATE = 0xB4)		Main flash	
		Nonsecure page	**Secure page**
Secure	Fetch	BUS ERROR	OK
	Read	RAZ, ILAFM	
	Write	WI, WRPERR, ILAFM	If not WRP: OK else: WI, WRPERR
	Erase		
Nonsecure	Fetch	OK	BUS ERROR
	Read		RAZ, ILAFM
	Write	If not WRP: OK else: WI, WRPERR	WI, WRPERR, ILAFM
	Erase		

Table 53. TZ protection and bank or mass erase summary

TZ protection (TZ_STATE = 0xB4)	**Main flash**		
	Nonsecure flash	**Secure flash**	**Mix of secure and nonsecure**
MES	WI, WRPERR, ILAFM	OK	WI, WRPERR, ILAFM
MENS	OK	WI, WRPERR, ILAFM	WI, WRPERR, ILAFM

Deactivated TrustZone® security

Reversing the TZ setting in OB (TZEN) is possible when the product state is Open or Provisioning. Deactivating TZEN from other states requires full regression (Regression state).

When TrustZone® is deactivated (TZ_STATE = 0xC3) after option bytes loading, the following security features are deactivated:

- Watermark-based secure area (refer to *Watermark-based secure flash area protection*)
- Block-based secure area (refer to *Section 7.6.3: Block-based secure flash memory area protection*)
- Secure interrupts (refer to *Section 7.9.11: Illegal access (ILAFM/ILAP)*)
- All secure registers are RAZ/WI.

Watermark-based secure flash area protection

When TrustZone® security is active (TZ_STATE = 0xB4), a part of the flash memory can be protected from nonsecure read and write access operations. Up to two different nonvolatile secure areas can be defined by option bytes and can be read or written by a secure access operation only: one area per bank can be selected with a page granularity.

The secure areas are defined by a start page offset and end page offset using the respective STRT and END option bytes. These offsets are defined in the secure watermark registers address registers *FLASH security watermark for Bank1 (FLASH_SECWM1R_CUR)*, *FLASH security watermark for Bank2 (FLASH_SECWM2R_CUR)* and modified by corresponding FLASH_XXX_PRG registers.

The STRT and END option bytes can only be modified by secure firmware.

Table 54. Secure watermark-based area

Secure watermark option bytes values (x = 1,2)	Secure watermark protection area
STRT > END	No secure area
STRT = END	One page defined by STRT is secure watermark-based protected
STRT < END	The area between STRT and END is secure watermark-based protected

Caution: Switching a memory area from secure to nonsecure does not erase its content. The user secure software must perform the needed operation to erase the secure area before switching an area to nonsecure whenever needed. It is also recommended to flush the instruction cache.

7.6.2 Hide protection (HDP)

The HDP area is independent of the flash watermark-based secure area. One nonvolatile secure hide protection (HDP) area per bank can be defined with a page granularity. Access to the hide protection area can be denied by progressing the HDPL level in the SBS.

When the HDPL = 1, no user flash memory is protected by the HDP. With HDPL ≥ 2 the user-OB-defined HDP area in each bank is closed. No read, write, fetch, or erase operation is allowed in the HDP area.

The HDP area can be extended. The extension setting is only a register value, not stored in OB. With HDPL = 3, the HDP area remains activated and the extension is added. If extension is added while the base user-OB-defined area is inactive, the extension covers one more page, because the END page is also protected.

The HDPL level can only be cleared by a system reset; there is no means to deactivate the HDP area.

The protected HDP area is defined by setting its size using start and end pages in a similar way as the secure watermark.

The size of the inaccessible area can be extended using register FLASH_HDPEXTR values HDPx_EXT. The value HDPx_EXT is a number of pages added to the HDP area (past the END page). The volatile HDPx_EXT value in the register can be increased but not decremented.

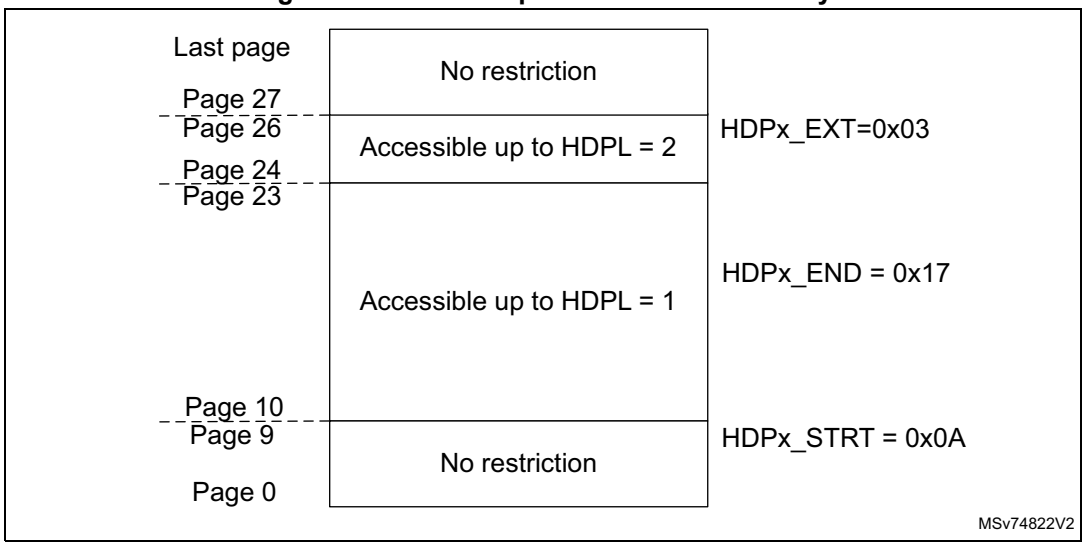
In case of STRT > END, the extension covers the (HDPx_EXT + 1) size area between pages marked by END and END + HDPx_EXT.

By default (from production), END is set to 0 and STRT is set to 1. This means that the HDP area in the user flash memory is size zero and extends from the flash memory start address (first page).

For example, to protect area by HDP from the address 0x0801 4000 (included) to the address 0x0802 FFFF (included):

- For physical Bank1, the FLASH_HDP1R must be programmed with:
  - STRT = 0x0A
  - END = 0x17

Figure 29. HDP example in user flash memory



- Once the HDPL is incremented to 2, the marked area becomes inaccessible. The HDPL = 2 code then sets the extension size to 3. Extension becomes inaccessible after HDPL is incremented to 3.
- Alternatively, in HDPL3 if:
  - END = 0, START = 1, HDPx_EXT = 1 page 0 and page 1 are HDP-protected.
  - END = 2, START = 7, HDPx_EXT = 3, pages 2 to 5 are HDP-protected (rather unusual example).
- If the two banks are swapped, the protection defined for physical Bank1 remains on the physical Bank1, unaffected by the swap. Separate protection applies to physical Bank2 and FLASH_HDP2R must also be programmed with:
  - STRT = 0x0A
  - END = 0x17

Note: For more details on the bank swapping mechanism, refer to Section 7.6.6: Flash memory bank attributes in case of bank swap.

Table 55. Secure hide protection

HDPx watermark option-byte values (x = 1, 2)		Hide protection area
HDPL ≤ 1	-	No inaccessible HDP area in user flash memory
HDPL = 2	END < STRT	No inaccessible HDP area in user flash memory
	END = STRT	Exactly one page is protected by HDP.
	END > STRT	The area between STRT and END is HDP protected.

Table 55. Secure hide protection (continued)

HDPx watermark option-byte values (x = 1, 2)		Hide protection area
HDPL = 3	END < STRT and HDPx_EXT = 0	No inaccessible HDP area in user flash memory
	END ≥ STRT or HDPx_EXT > 0	The area between min(STRT, END) and (END + HDPx_EXT) is HDP-protected.

Table 56. HDP protections summary

HDP protection	User flash			
	Access to OB HDP area		Access to EXT HDP area	
	HDPL 2 or 3	HDPL1	HDPL3	HDPL1or HDPL 2
Fetch	BUS ERROR	OK	BUS ERROR	OK
Read	RAZ		RAZ	
Write	WI, WRPERR	If not WRP: OK else: WI, WRPERR	WI, WRPERR	If not WRP: OK else: WI, WRPERR
Erase (mass erase)				

7.6.3 Block-based secure flash memory area protection

Any page can be programmed on-the-fly as secure or nonsecure using the block-based configuration registers. FLASH_SECBB1/FLASH_SECBB2 registers are used to configure the security attribute for pages in Bank1/Bank2 respectively.

When the page security attribute bit SECy[i] is set, the security attribute is the same as the secure watermark-based area. The secure page is accessible only by a secure access operation.

If the SECy[i] bit is set or reset for a page already included in a secure watermark-based area, the page keeps the watermark-based protection security attributes.

To modify a block-based page security attribute, it is recommended to:

- Check that no flash operation is ongoing on the related page.
- Add an ISB instruction after modifying the page security attribute SECy[i].

Caution: Switching a page from secure to nonsecure does not erase the content of the associated page. User secure software must perform the needed operations before switching to the nonsecure attribute:

- Erase page content.
- Invalidate the instruction cache.

Note: For SECy[i] access control, refer to Table 57.

Execution protection for the S-Bus (AHB register bus) is done in AHB decoder, not in the flash memory interface itself.

Table 57. Secure configuration block-based registers access conditions

Access			Corresponding page privileged status	Page setting in SECBBxR
Fetch	Secure/ nonsecure	Privileged/ unprivileged	-	BUS ERROR
Read			-	OK
Write	Secure	Privileged	-	OK
		Unprivileged	0	OK
			1	WI
	Nonsecure	Privileged/ unprivileged	-	WI, ILAP

7.6.4 Block-based privileged flash memory area protection

Any page can be programmed on the fly as privileged or unprivileged using the block-based configuration registers. The FLASH_PRIVBB1 and FLASH_PRIVBB2 registers are used to configure the privilege attribute for pages in Bank1/Bank2 respectively.

When the page privilege attribute PRIVy[i] bit is set, the page is accessible only for privileged access. An unprivileged page is accessible by privileged or unprivileged access.

To modify a block-based privilege attribution, it is recommended to:

- Check that no flash operation is ongoing on the related page.
- Add an ISB instruction after modifying the page security attribute PRIVy[i].

Caution: Switching a page from privileged to unprivileged does not erase the content of the associated page.

Table 58. Privilege protection summary

Access (TZ not relevant)		Main flash	
		Unprivileged page	Privileged page
Privileged	Fetch	OK	
	Read		
	Write		
	Erase		
Unprivileged	Fetch	OK	RAZ
	Read		
	Write		WI, WRPERR
	Erase		

Table 59. Privilege and mass or bank erase

Access (TZ state not relevant)	Main flash		
	Unprivileged FLASH	Privileged FLASH	Mix unprivileged and privileged FLASH
MEP	OK		
MENP	OK	WI, WRPERR	

Note: For PRIVy[i] access control, refer to Table 60 and Table 61.

Table 60. Privilege configuration register access conditions (TZ enabled)

Access			Corresponding page secure status(1)	Page setting access in PRIVBByR
Fetch	Privileged/ unprivileged	Secure/ nonsecure	-	BUS ERROR
Read			-	OK (all pages)
Write	Privileged	Secure	-	OK (all pages)
		Nonsecure	NS	OK (only the page corresponding to the bit)
			S	WI (only the page corresponding to the bit)
	Unprivileged	Secure/ nonsecure	-	WI

1. Either watermark or block-based.

Table 61. Privilege configuration register access conditions (TZ disabled)

Access		Page setting access in PRIVBByR
Fetch	Privileged/ unprivileged	BUS ERROR
Read		OK
Write	Privileged	OK
	Unprivileged	WI

7.6.5 Flash memory register privileged and unprivileged modes

The flash registers can be read and written by privileged and unprivileged access, depending on the SPRIV and PRIV bits in *FLASH privilege configuration register (FLASH_PRIVCFGR)*:

- When the SPRIV/PRIV bit is reset, all secure/nonsecure) flash registers can be read and written by both privileged or unprivileged access.
- When the SPRIV/PRIV bit is set, all secure/nonsecure flash registers can be read and written by privileged access only. Unprivileged access to a privileged register is RAZ/WI.

The registers related to key storage (FLASH_OBKCFGR, FLASH_SOBKCFGR, FLASH_OBKKEYR, FLASH_SOBKEYR) are exceptions to this rule and can only be accessed in privileged mode, regardless of other settings.

The table below summarizes flash registers access control.

Table 62. Flash register accesses(1)

Access			Nonsecure register		Secure register	
			PRIV = 1	PRIV = 0	SPRIV = 1	SPRIV = 0
X	Secure/ nonsecure	Privileged/ unprivileged	Bus error			
R/W	Secure	Privileged	OK			
R/W		Unprivileged	RAZ, WI	OK	RAZ, WI	OK
R/W	Nonsecure	Privileged	OK		RAZ, WI, ILAP	
R/W		Unprivileged	RAZ, WI	OK		

1. Some registers are privilege only and not configurable.

7.6.6 Flash memory bank attributes in case of bank swap

The FB_MODE setting in SBS modifies the address of each bank in the memory map. When not active, flash Bank1 is mapped at the lower address range. FB_MODE reset value is controlled by the SWAP_BANK option byte, controlled in FLASH_OPTSR3. When FB_MODE is set, flash Bank1 is mapped at the higher address range. Flash bank attributes follow their bank contents so there is no need to modify their setting registers when swapping banks:

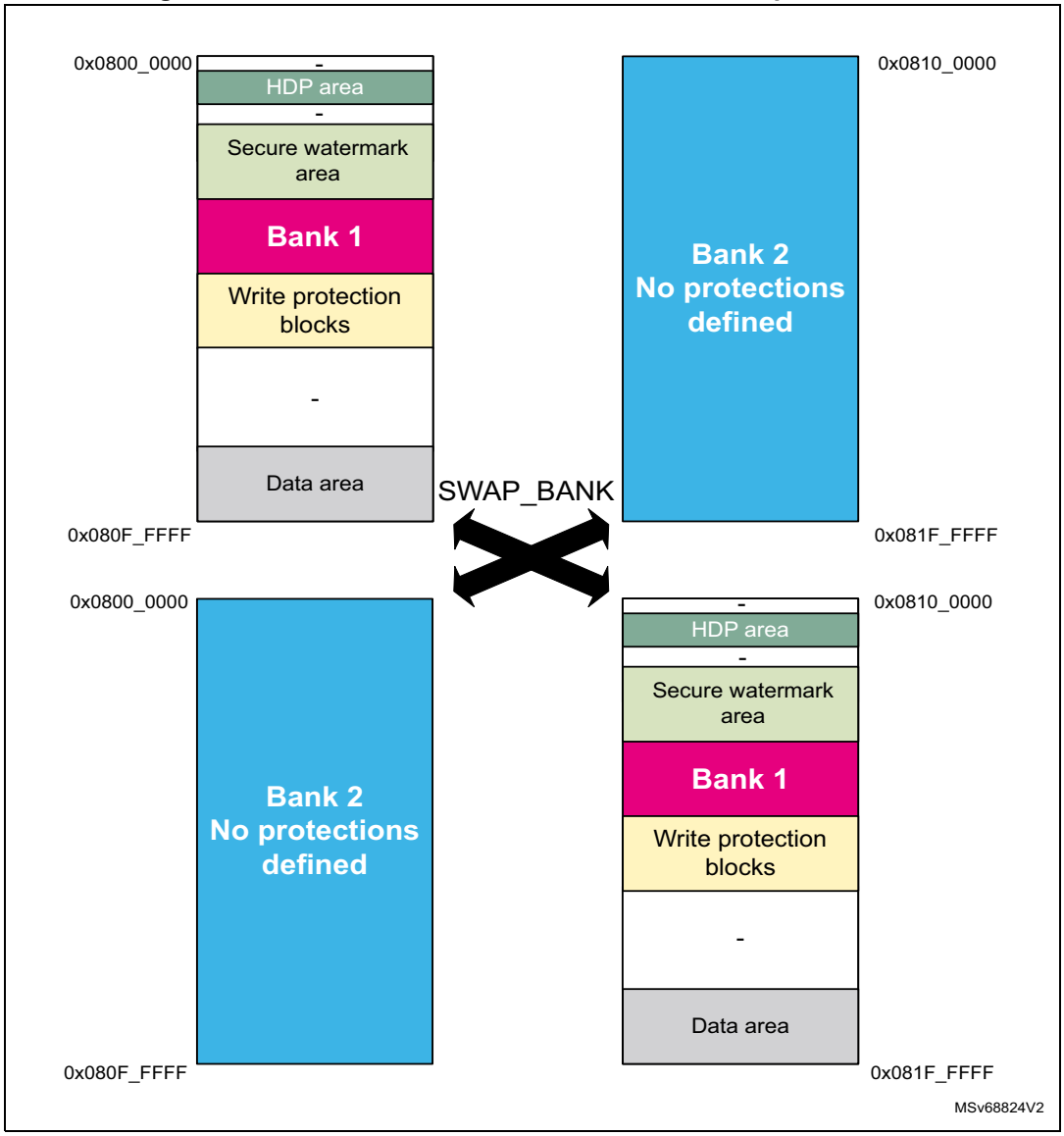
- Flash secure watermark x FLASH_SECWMx
- Flash write protection page group FLASH_WRPG (refer to *Section 7.6.8: Write protection*)
- Hide protection in FLASH_HDPx registers
- Flash secure block based bank x register FLASH_SECBBxR
- Flash privilege block based bank x register FLASH_PRIVBBxR
- The data area configuration FLASH_EDATAx

The SWAP_BANK is rendered immutable by setting SWAP_LOCKOUT in the user OB. This sets the reset value for SWAP_LOCK in the SBS that locks the SWAP_BANK option.

Note: The BK_ECC bit in the FLASH ECC detection register (FLASH_ECCDETR) and FLASH ECC correction register (FLASH_ECCCORR), BKSEL bit in FLASH nonsecure control register (FLASH_CR) and BKSEL bit in FLASH secure control register (FLASH_SCR) always refers to Bank1 (respectively Bank2) when it is low (respectively high), regardless of the SWAP_BANK value.

Figure 30 below shows how security attributes and protections behave in case of bank swap.

Figure 30. Protection attributes in case of bank swap illustration



7.6.7 Flash memory configuration protection

The memory uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (such as software bugs):

- Option bytes change
- Write operations
- Erase commands
- Interrupt masking

The memory configuration registers protection is summarized in *Table 63*. Registers not present in this table are not protected by a key.

Table 63. Flash interface register protection summary

Register name	Unlocking register	Protected asset
FLASH_CR	FLASH_KEYR	Write/erase control
FLASH_SCR	FLASH_SKEYR	Secure write/erase control
FLASH_OPTCR + all _PRG registers	FLASH_OPTKEYR	Flash bank option byte words change
FLASH_OBKCFGR	FLASH_OBKKEYR	Flash OBKey storage manipulation
FLASH_SOBKCFGR	FLASH_SOBKKEYR	

7.6.8 Write protection

The purpose of write protection is to prevent unwanted modifications to code and/or data.

Any flash group of four consecutive 8-Kbyte pages can be independently write-protected or unprotected by clearing/setting the corresponding WRPGn1/2 bit in the FLASH_WRPG1/2 register (see *FLASH write page group protection for Bank1 (FLASH_WRP1R_CUR)* and *FLASH write page group protection for Bank2 (FLASH_WRP2R_CUR)*).

A write-protected group of pages cannot be erased nor programmed. As a result, a bank erase cannot be performed if one group of pages is write-protected, unless an PRODUCT_STATE transition to OPEN is triggered (erasing the entire user flash memory).

Note: Write protection errors are documented in Section 7.9: FLASH error management.

7.6.9 Flash high-cycle data protections

The embedded flash memory can be configured to have up to 96-Kbyte memory area with high-cycling capability (100 kcycles) to store data and emulate EEPROM (see *Section 7.3.11*).

Option-byte controlled security features for user flash memory prevent pages from being used as high-cycle data area. No OPTCHANGEERR is raised in case of setting conflict, it only makes the pages unusable. Details are listed in *Table 64*.

Volatile settings using FLASH_SECBBxR and FLASH_PRIVBBxR can be applied. In this case the security setting of user flash page 63 applies to flash high-cycle data page 0 and so on, as indicated in *Figure 19*, *Figure 24*, and *Figure 25*.

Volatile settings of HDP extension that extends into pages of data area causes the data area to be inaccessible.

Table 64. High-cycle area protection summary: access to data area address range

<table>
<tr><th colspan="2" rowspan="3">TZ protection (when TZ_STATE = 0xB4)(1)</th><th colspan="6">Data page enabled</th><th rowspan="3">Data page not enabled</th></tr>
<tr><th rowspan="2">HDP protected</th><th rowspan="2">OB secure area</th><th colspan="2">SECBBxR protected</th><th colspan="2">NS page</th></tr>
<tr><th>16/32 bits access</th><th>Other</th><th>16/32 bits access</th><th>Other</th></tr>
<tr><td rowspan="4">Secure</td><td>Fetch</td><td colspan="2" rowspan="3">BUS ERROR</td><td colspan="4">BUS ERROR</td><td rowspan="5">BUS ERROR</td></tr>
<tr><td>Read</td><td>OK</td><td rowspan="2">BUS ERROR</td><td colspan="2">RAZ, ILAC</td></tr>
<tr><td>Write</td><td>If not WRP: OK else WI, WRPERR</td><td colspan="2">WI, WRPERR, ILAFM</td></tr>
<tr><td>Erase</td><td colspan="6">Same protection as in corresponding classic user flash page</td></tr>
<tr><td rowspan="4">Nonsecure</td><td>Fetch</td><td colspan="2" rowspan="3">BUS ERROR</td><td colspan="4">BUS ERROR</td></tr>
<tr><td>Read</td><td colspan="2">RAZ, ILAC</td><td>OK</td><td rowspan="2">BUS ERROR</td></tr>
<tr><td>Write</td><td colspan="2">WI, WRPERR, ILAFM</td><td>If not WRP: OK else WI, WRPERR</td></tr>
<tr><td>Erase</td><td colspan="6">Same protection as in corresponding classic user flash page</td></tr>
</table>

1. In case of TZ_STATE = 0xC3, there is no secure state access, no secure BB and no Secure Area. For HDP and NS access the rules remain the same.

Table 65. HDP protected definition

User flash	OB HDP Area enabled	HDP extension enabled
HDPL1	HDP protected = 0	HDP protected = 0
HDPL2	HDP protected = 1	HDP protected = 0
HDPL3	HDP protected = 1	HDP protected = 1

Table 66. Privilege pages and the data area - access to data area address range

<table>
<tr><th colspan="2" rowspan="2">Access</th><th colspan="2">Data area address range access</th></tr>
<tr><th>Unprivilege page</th><th>Privilege page</th></tr>
<tr><td rowspan="3">Privilege</td><td>Fetch</td><td colspan="2">BUS ERROR</td></tr>
<tr><td>Read</td><td>OK</td><td>OK</td></tr>
<tr><td>Write</td><td>OK</td><td>OK</td></tr>
<tr><td rowspan="3">Unprivilege</td><td>Fetch</td><td colspan="2">BUS ERROR</td></tr>
<tr><td>Read</td><td>OK</td><td>RAZ, ILAC</td></tr>
<tr><td>Write</td><td>OK</td><td>WI, WRPERR</td></tr>
</table>

7.6.10 Life cycle management

Nonvolatile or debug states are determined by the product state set by user option bytes. Debug possibility and the possibility to change selected security settings are related to it. While the state is stored in the OB, it is the SBS that controls the debug policy.

OBK storage access conditions are unaffected by the PRODUCT_STATE, but keys can be evicted in regression transitions.

HDP works equally regardless of the debug state. Even in debug the HDP area is hidden when HDPL increments.

The following states are defined:

Table 67. Product states, debug states, and debug policy

PRODUCT_STATE	Code in OB	Description
Open	0xED	User flash open (TZ secure and nonsecure open). External access (debug) N and NS enabled (~RDP0).
Provisioning	0x17	Provisioning - immutable root of trust is being installed. Open if TZEN = 0xC3.
iRoT-Provisioned	0x2E	The immutable root of trust is installed (~RDP 0.5).
TZ-Closed	0xC6	State for debugging the NS application. Debug is restrained to nonsecure areas.
Closed	0x72	State for running a secure application. Debug disabled, regression is possible.
Locked	0x5C	Transition to other state, policy change or debug not permitted(1).
Regression	0x9A	The temporary state initiated by the debug authentication system in transition to Open.
NS-Regression	0xA3	The temporary state initiated by the debug authentication system in transition to TZ-Closed.

1. The same situation is Closed state with incorrect or undefined debug certificates.

1. No debug protection

Read, program, and erase operations into the main flash memory area are possible. Least restriction on the option bytes.

2. Nonsecure debug only

All read and write operations (if no write protection is set) from/to the nonsecure flash memory are possible. Debug access to the secure area is prohibited. Debug access to the nonsecure area remains possible.

The following rules are enforced:

- User mode: code executing in user mode (boot flash) can access the main flash memory and option bytes with all operations (read, erase, program).
- Nonsecure debug mode: the secure flash memory areas are inaccessible; the nonsecure flash memory areas remain accessible for debug purposes.

- Boot RAM mode: boot from SRAM is not possible.
- Transition to Open state is possible with the consequence of a flash memory mass erase and key slot revocation, depending on the debug unlock policy (see below).

3. Full protection

All debug features are disabled and the following rules enforced:

- User mode: code executing in user mode (boot flash) can access the main flash memory and selected option bytes with all operations (read, erase, program).
- Boot RAM mode: boot from SRAM is not possible.
- Transition to TZ-Closed or Open state is possible with the consequence of a flash memory mass erase and key slot revocation, depending on the debug unlock policy (see below).

7.6.11 Product state transitions

Progressing to a more closed state is the normal product life cycle, that does not require security measures. Transitioning to an open state is a regression, controlled by the debug authentication control. If the debug unlock policy is set to “locked”, no regression is accepted. The locked state can be reinforced by invalidating the debug authentication certificates.

All transitions not listed in this chapter are invalid.

There is no restriction on reading the current product state.

Transitions in progress direction

The transition from Open to any of the closed states is a matter of correct product configuration and provisioning. The transitions must be done in the correct order.

The normal progressing order is:

- Open, Provisioning, or iRoT-Provisioned
- Provisioning to iRoT-Provisioned
- iRoT-Provisioned to Closed, Locked, or Closed
- Closed to Locked or Closed

The transition is managed either by software running on the device, or directly, using a debug interface. Transitions from Closed are only possible by software running on the device. By software it is assumed that transitions are triggered by the bootloader, or root-of-trust services, but generally any software running on the device can do a progression transition.

Transition to Open state

This transition is a full regression. The starting state can be any except Open and Locked. The debug tools are used to authenticate debug regression access rights with the debug authentication library, running on the device in HDPL1 (refer to *Section 14.3.7: SBS debug control*). After verifying the credentials, it puts the device in the intermediate Regression state. From this state, the device regresses securely to Open.

The transition has the following consequences:

- Epoch secure counter incremented
- Key slots revocation
- Product state updated
- User options updated (security settings reset, TZEN reset)
- Flash memory mass erase

Transition from closed to TZ-Closed debug state

This transition opens the device to authorize the debugging of the nonsecure application software without compromising the security of the RoT functions.

The starting state is Closed. The debug tools are used to authenticate debug regression access rights with the debug authentication library, running on the device in the HDPL1 (refer to *Section 14.3.7: SBS debug control*). After verifying the credentials, it puts the device into intermediate state NS-Regression. From this state, the device regresses securely in Closed.

The transition has the following consequences:

- Epoch nonsecure counter incremented
- Nonsecure slot revocation
- Product state updated
- Flash memory pages that are not covered by the secure watermark are erased

Table 68. PRODUCT_STATE transitions

From	To[1]							
	Open	Provisioning	iRoT-Provisioned	TZ-Closed[2]	Closed	Locked	Regression	NS-Regression[2]
Open	-	OK	OK	X	X	X	X	X
Provisioning	X	-	OK	OK	OK	OK	OK (HDPL1)	X
iRoT-Provisioned	X	X	-	OK	OK	OK	OK (HDPL1)	X
TZ-Closed	X	X	X	-	OK	OK	OK (HDPL1)	X
Closed	X	X	X	X	-	X	OK (HDPL1)	OK (HDPL1)
Locked	X	X	X	X	X	-	X	X
Regression	OK	X	X	X	X	X	-	X
NS-Regression	X	X	X	OK	X	X	X	-

1. X = transitions not permitted (OPTCHANGEERR raised), - = no change, OK = valid transitions.
2. To transition into this state, both current and programmed value of TZEN must be enabled (TZEN = 0xB4), otherwise the change is ignored with OPTCHANGEERR.

In *Table 68*, X indicates transitions that are not permitted, a dash indicates no change, and OK indicates valid transitions. Some transitions are possible only in one specific HDPL, provided in parenthesis. Attempts at an illegal transition result in OPTCHANGEERR.

An incorrect PRODUCT_STATE is interpreted as Locked.

7.6.12 OBK protection

At the receipt of an OBK access (read/write/execute), secure and privileged attributes are first checked, followed by OBK-HDPL. The OBK-HDPL value must exactly match the HDPL assigned to the key location, otherwise the OBKERROR flag is raised. The flash memory interface response is described in *Table 69*.

Table 69. TZ OBK protection summary

TZ protection (when TZ_STATE = 0xB4)	OBK access		
	No tamper		Tamper
	HDPL1/2/3	OBK selector	
XS	BUS ERROR	BUS ERROR	RAZ, WI
RS	OK if privileged, otherwise RAZ	RAZ	
WS	OK if privileged, otherwise WI, WRPERR	WI, if not privileged also WRPERR	
XNS	BUS ERROR	BUS ERROR	
RNS	RAZ, ILAFM	RAZ, ILAFM	
WNS	WI, WRPERR, ILAFM	WI, WRPERR, ILAFM	
Erase	NA	NA	NA

Table 70. OBK protection summary with TZ disabled

TZ protection (when TZ_STATE = 0xC3)	OBK access		
	No tamper		Tamper
	HDPL1/2/3	OBK selector	
XNS	BUS ERROR	BUS ERROR	RAZ, WI
RNS	OK if privilege, otherwise RAZ	RAZ	
WNS	OK if privilege, otherwise WI, WRPERR	WI if privilege, otherwise WI, WRPERR	
Erase	NA	NA	NA

Note: Cacheable access is managed at system level and not in the flash memory interface. The MPU must be configured to disable cache where it is not desirable.

In case of a tamper, all keys are read as 0x00.

There are also rules for accessing the OBKFGCR and OBKKEYR registers.

Table 71. Access conditions to secure control register

-	S/NS	P/NP	TZ state	SOBKCFGR and SOBKKEYR access
X	-	-	-	Bus error
R/W	S	P	Active	OK
R/W	S	NP	Active	RAZ, WI
R/W	NS	P	Active	RAZ, WI, ILAP
R/W	NS	NP	Active	RAZ, WI, ILAP
R/W	-	-	Deactivated	RAZ, WI

Table 72. Access conditions to nonsecure control register

	S/NS	P/NP	TZ state	OBKCFGR and OBKKEYR access
X	-	-	-	Bus error
R/W	-	-	Active	RAZ, WI
R/W	-	P	Deactivated	OK
R/W	-	NP	Deactivated	RAZ, WI

7.6.13 One-time-programmable and read-only memory protections

Sections of OTP/RO in the flash memory are described in detail in *Section 7.3.10*. There is no protection provided by the flash memory interface other than the dedicated write protection.

No write is possible to the RO area. *OTP write protection* describes a method of locking out the OTP area.

The access conditions are summarized in *Table 73*.

Table 73. OTP/RO access constraints

-	8 Kbytes page dedicated to RO/OTP			
	0x08FF_E000 - E7FF	**0x08FF_E800 - EFFF Reserved**	**0x08FF_F000 - F7FF OTP**	**0x08FF_F800 - FFFF RO**
XS	BUS ERROR	BUS ERROR		
RS		RAZ, ILAC		
WS		WI, WRPERR, ILAFM		
XNS	BUS ERROR	BUS ERROR		
RNS		If correct size(1) RAZ, else BUS ERROR	If correct size OK, else BUS ERROR	
WNS		If correct size WI, else BUS ERROR	BUS ERROR if incorrect size. WRPERR if block is protected by LOCKBL, else OK.	If correct size WI, else BUS ERROR
Erase	WI			

1. Word size is 16 bits. 32-bit access is possible. Any other access attempts lead to bus error.

7.7 System memory

7.7.1 Introduction

System memory stores RSS (root secure services) firmware programmed by STMicroelectronics during production. RSS provides runtime services to user firmware.

When TrustZone® is enabled (user sets the TZEN bitfield to 0xB4), RSS provides secure services to the secure user firmware only. When TrustZone® is disabled (user sets the TZEN bitfield to 0xC3), RSS provides nonsecure services to the user firmware.

7.7.2 RSS user functions

The RSS provides runtime services using the RSS library, whose functions are exposed to the user within the CMSIS device header file, provided by the STM32Cube firmware package.

RSS provides services through two different libraries, RSSLIB and NSSLIB, dedicated to secure user firmware when the system is configured, respectively, with TrustZone® enabled (TZEN bitfield set to 0xB4) and TrustZone® disabled (TZEN bitfield set to 0xC3).

Table 74. Macros for RSS services

TrustZone	C defined macro	Location in flash memory
Enabled	RSSLIB_PFUNC	0x0BFADB68
Disabled	NSSLIB_PFUNC	0x0BFADB6C

RSSLIB

The secure and nonsecure firmware can call RSSLIB functions:

- Secure firmware can call secure callable functions and nonsecure callable functions.
- Nonsecure firmware can only call nonsecure callable functions.

The user firmware calls the RSSLIB functions using RSSLIB_PFUNC C-defined macro, which points to a location within the nonsecure system memory. Before calling the RSSLIB functions, the secure user firmware must:

- Define a nonsecure region above this location within SAU of the Cortex®-M33 processor, starting from RSSLIB_SYS_FLASH_NS_PFUNC_START (0x0BFADB68) up to RSSLIB_SYS_FLASH_NS_PFUNC_END (0x0BFADB84).
- Define a nonsecure calleable region above the RSSLIB-NSC location within SAU of the Cortex®-M33 processor, starting from RSSLIB_NSC_BASE (0x0FF9E040) up to RSSLIB_NSC_END (0x0FF9E13F).
- Ensure that the RSSLIB is executable. The RSSLIB is located in the secure system flash address RSSLIB_BASE (0x0FF9E040) with a size of RSSLIB_SIZE (0x3FC0). These addresses and size are provided within the CMSIS device header file.

The user can configure the SAU regions either by using the CMSIS system partition header file, or by implementing their own code for SAU setup. The CMSIS system partition header file is part of the STM32Cube firmware package.

Table 75. RSS lib interface functions

Library	Function	Attributes
RSSLIB_PFUNC	JumpHDPLvl2	Secure callable function
	JumpHDPLvl3	
	JumpHDPLvl3NS	
	RSSLIB_DataProvisioning	Nonsecure callable function

RSSLIB_DataProvisioning

Secure attribute: nonsecure callable function.

Prototype:

```
uint32_t RSSLIB_DataProvisioning_(RSSLIB_DataProvisioningConf_t *pConfig)
```

Arguments:

`pConfig`: input parameter. **`RSSLIB_DataProvisioningConf_t`**

The C structure definition is described below:

```
typedef struct
{
  uint32_t *pSource;
  uint32_t *pDestination;
  uint32_t Size;
  uint32_t DoEncryption;
  uint32_t Crc;
} RSSLIB_DataProvisioningConf_t;
```

Structure elements

`pSource` Provides the address of the data to be provisioned. Must be within the SRAM1 address range (nonsecure aliases).

`pDestination` Provides the address where to store the data to be provisioned. Must be within the OBKeys address range. To be aligned on 16 bytes.

`Size` Provides the size of the data to be provisioned (the number of bytes, must be a multiple of 16).

`DoEncryption` Notifies RSSLIB_DataProvisioning function if it must encrypt or not the data within OBKeys.

DoEncrypt can be either 0xF5F5A0AAU or 0xCACA0AA0U

– 0xF5F5A0AAU: notifies RSSLIB_DataProvisioning to encrypt data with relevant DHUK before programming it within OBKeys. DHUK is selected according to **`pDestination`** value. This value is allowed only on STM32G55x/56x devices.
– 0xCACA0AA0U: notifies RSSLIB_DataProvisioning to program in clear data within OBKeys.

`Crc` CRC over full source data buffer, **`pConfig->pDestination`** value, **`pConfig->Size`** value and finally **`pConfig->DoEncryption`** value.

CRC computation uses CRC-32 (Ethernet):

– CRC polynomial: 0x04C11DB7U
– Initial value: 0xFFFFFFFFU

Returned values

0xEAEAEAEAU	Success
0xF5F5F5F5U	Error: – **`pConfig`** or **`pSource`** are not in SRAM1
0xF5F5E0E0U	Error: – **`pDestination`** is not aligned on 16 bytes – **`pDestination + Size`** does not fit within an OBKey section
0xF5F50E0EU	Error: **`Size`** is not a multiple of 16
0xF5F58080U	Error: computed CRC is not the expected one provided within **`pConfig->CRC`**
0xF5F58008U	Error: cannot program data within OBKeys destination section
0xF5F5E00EU	Error: wrong **`DoEncryption`** parameter value
0xF5F50880U	Error: encryption requested, butw platform does not support it
0xF5F50EE0U	Error: encryption error
0xF5F50808U.	Error: OBKeys programming error

RSSLIB_DataProvisioning receives in input a data buffer and programs it within OBKeys. A CRC prevents any data and parameter tampering issues.

The destination address (**`pDestination`**) added to the size (**`Size`**) of the data to be provisioned must not cross the OBKeys section boundaries, listed below.

OBKeys Level1	Start address: 0x0FFD0100UL
	End address: 0x0FFD08FFUL
OBKeys Level2	Start address: 0x0FFD0900UL
	End address: 0x0FFD0FFFUL
OBKeys Level3 secure	Start address: 0x0FFD1000UL
	End address: 0x0FFD17FFUL
OBKeys Level3 nonsecure	Start address: 0x0FFD1800UL
	End address: 0x0FFD1FEFUL

When requested through the **`pConfig->DoEncryption`** parameter, RSSLIB_DataProvisioning encrypts the data before programming them within OBKeys.

RSSLIB_DataProvisioning uses AES CBC 128 bits with:

- IV: defined as (using C definition format)

  `uint32_t IV = {0x8001D1CEU, 0xD1CED1CEU, 0xD1CE8001U, 0xCED1CED1U};`
- Key: DHUK corresponding to the targeted OBKeys section defined by **`pConfig->pDestination`** parameter.

Note: Data encryption within the OBK is only supported on devices with a hardware cryptographic accelerator.

Note: This function uses the following peripherals:

- CRC, RNG, SAES, HASH, PKA, CCB: reset, use and reset
- SBS, FLASH, GTZC: use

JumpHDPLvl2

Secure attribute: Secure callable function.

Prototype:

```
uint32_t JumpHDPLvl2(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
RSSLIB_PFUNC->S.JumpHDPLvl2((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex®-M33 processor.
- MPUIndex:
  - Input parameter, MPU region index. The caller function must define (but keep disabled) the corresponding MPU region before calling JumpHDPLvl2. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

The user calls JumpHDPLvl2 to close the user flash memory HDPL1 area by incrementing HDPL to 2, then jump to the reset handler embedded within the vector table, whose address is passed as an input parameter.

After closing HDPL1, JumpHDPLvl2 enables the MPU region provided as an input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table, and jumps to the reset handler function supported by it. JumpHDPLvl2 does not set the new vector table. To ensure the correct interrupts handling during this service, the user must choose one of the following options:

- option 1: the user disables the interrupts before calling JumpHDPLvl2 and restores the interrupts once in HDPL2 after setting the new vector table
- option 2: the user ensures the vector tables are always accessible (i.e. vector tables are located outside the HDPL areas)

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), RSSLIB_Sec_JumpHDPLvl2 returns 0xF5F5F5F5.

JumpHDPLvl3

Secure attribute: secure callable function.

Prototype:

```
uint32_t JumpHDPLvl3(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
RSSLIB_PFUNC->S.JumpHDPLvl3((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex®-M33 processor.
- MPUIndex:
  - Input parameter, MPU region index. Caller function must define but keep disabled the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the reset handler of the vector table, whose function belongs to the MPU region.

User calls JumpHDPLvl3 to close the user flash memory HDPL1 and HDPL2 areas by incrementing HDPL up to 3, then jump to the reset handler embedded within the vector table, whose address is passed as input parameter.

After closing HDPL1/2, JumpHDPLvl3 enables the MPU region provided as an input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table, and jumps to the reset handler function supported by the vector

table. JumpHDPLvl3 does not set the new vector table. To ensure the correct interrupts handling during this service, the user must choose one of the following options:

- option 1: the user disables the interrupts before calling JumpHDPLvl3 and restore the interrupts once in HDPL3 after setting the new vector table
- option 2: the user ensures the vector tables are always accessible (i.e. vector tables are located outside the HDPL areas)

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), JumpHDPLvl3 returns 0xF5F5F5F5.

JumpHDPLvl3NS

Secure attribute: secure callable function.

Prototype:

```
uint32_t JumpHDPLvl3NS(uint32_t VectorTableAddr)
```

User code function call example:

```
RSSLIB_PFUNC->S.JumpHDPLvl3NS((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex®-M33 processor.

User calls JumpHDPLvl3NS to close the user flash memory HDPL1 and HDPL2 areas by incrementing HDPL up to 3, to move from secure to nonsecure domain, then jump to the nonsecure reset handler embedded within the vector table, whose address is passed as an input parameter.

After closing HDPL1/2, JumpHDPLvl3 jumps to the nonsecure reset handler function supported by the vector table. JumpHDPLvl3NS sets the new nonsecure vector table and the new nonsecure SP to the address provided by the passed vector table. To ensure the correct interrupts handling during this service, the user must choose one of the following options:

- option 1: the user disables the interrupts before calling JumpHDPLvl3NS, configure NVIC to allow selected nonsecure IRQ and restore the interrupts once in HDPL3NS
- option 2: the user ensures the vector tables are always accessible (i.e. vector tables are located outside the HDPL areas)

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), JumpHDPLvl3NS returns 0xF5F5F5F5.

NSSLIB

The user firmware calls only the NSSLIB function when TrustZone is disabled (TZEN bitfield is set to 0xC3).

The user firmware calls NSSLIB functions using NSSLIB_PFUNC C defined macro, which points to a location within system memory. The user firmware must:

- Ensure that the NSSLIB is executable. The NSSLIB is located in nonsecure system flash address NSSLIB_BASE (0x0FFA2150) with a size of NSSLIB_SIZE (0xAB0).

These address and size are provided within the CMSIS device header file.

Table 76. NSS lib interface functions

Library	Function	Attributes
NSSLIB_PFUNC	JumpHDPLvl2	Nonsecure function
	JumpHDPLvl3	

JumpHDPLvl2

Prototype:

```
uint32_t JumpHDPLvl2(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
NSSLIB_PFUNC->JumpHDPLvl2((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex®-M33 processor.
- MPUIndex:
  - Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl2. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

User calls JumpHDPLvl2 to close the user flash memory HDPL1 area by incrementing HDPL to 2, and to jump to the reset handler embedded within the vector table which address is passed as an input parameter

After closing HDPL1, JumpHDPLvl2 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table and jumps to the reset handler function supported by the vector table too. JumpHDPLvl2 does not set the new vector table. To ensure the correct interrupts handling during this service, the user must choose one of the following options:

- option 1: the user disables the interrupts before calling JumpHDPLvl2 and restore the interrupts once in HDPL2 after setting the new vector table
- option 2: the user ensures the vector tables are always accessible (i.e. vector tables are located outside the HDPL areas)

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), RSSLIB_Sec_JumpHDPLvl2 returns 0xF5F5F5F5.

JumpHDPLvl3

Prototype:

```
uint32_t JumpHDPLvl3(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
NSSLIB_PFUNC->JumpHDPLvl3((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex®-M33 processor.
- MPUIndex:
  - Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

User calls JumpHDPLvl3 to close the user flash memory HDPL1 and HDPL2 areas by incrementing HDPL up to 3, and then jump to the reset handler embedded within the vector table, whose address is passed as an input parameter.

After closing HDPL1/2, JumpHDPLvl3 enables the MPU region provided as an input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table, and jumps to the reset handler function supported by it. JumpHDPLvl3 does not set the new vector table. To ensure the correct interrupts handling during this service, the user must choose one of the following options:

- option 1: the user disables the interrupts before calling JumpHDPLvl3 and restore the interrupts once in HDPL3 after setting the new vector table
- option 2: the user ensures the vector tables are always accessible (i.e. vector tables are located outside the HDPL areas)

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), JumpHDPLvl3 returns 0xF5F5F5F5.

7.8 FLASH low-power modes

Table 77 summarizes the memory behavior in STM32 low-power modes. The embedded flash memory belongs to the core domain.

Table 77. Effect of low-power modes on the embedded flash memory

Power mode	Core domain voltage range	Allowed if FLASH busy	FLASH power mode
Run	VOS0/1/2	Yes	Run
Stop1 (clock stopped)	SVOS3	No	Clock gated
Standby	Off	No	Off

When the system state changes or within a given system state, the memory can operate with a different voltage supply range (VOS), according to the application. The procedure to switch the memory into power modes (run, clock-gated, stopped, off) is described hereafter.

Note: For more information on the microcontroller power states, refer to Section 10: Power control (PWR).

Managing the FLASH domain switching to Stop or Standby

As explained in *Table 77*, if the memory informs the reset and clock controller (RCC) that it is busy (BSY, DBNE, WBNE is set), the microcontroller cannot switch the core domain to Stop or Standby mode.

There are two ways to release the memory:

- Reset the WBNE busy flag in the FLASH_SR/SSR register by any of the following actions:
  - Complete the write buffer with missing data.
  - Force the write operation without filling the missing data by activating the FW bit in the FLASH_CR/SCR register. This forces all missing data to “high”.
- Poll BSY busy bits in FLASH_SR/SSR register until they are cleared. This indicates that all recorded write, erase, and option change operations are complete.

The microcontroller can then switch the domain to Stop or Standby mode.

7.9 FLASH error management

7.9.1 Introduction

The memory automatically reports when an error occurs during a read, program, or erase operation. A wide range of errors are reported:

- *Nonsecure write protection error (WRPERR)*
- *Secure write protection error (WRPERR)*
- *Nonsecure programming sequence error (PGSERR)*
- *Secure programming sequence error (PGSERR)*
- *Secure strobe error (STRBERR)*
- *Error correction code error (ECCC, ECCD)*
- *Illegal access (ILAFM/ILAP)*
- *Option-byte change error (OPTCHANGEERR)*
- OBK nonsecure general error (OBKERR)
- OBK secure general error (OBKERR)
- OBK nonsecure write error (OBKWERR)
- OBK secure write error (OBKWERR)

The application software can individually enable the interrupt for each error, as detailed in *Section 7.10: FLASH interrupts*.

The flash memory interface uses different interrupt lines to trigger the event. There are two direct lines to NVIC, one for secure interrupts, another for nonsecure. The third line leads to TZIC, for the ILAFM interrupt.

Note: For all errors, the application software must clear the error flag before attempting a new modify operation.

Since there is just one write buffer and only one operation allowed at a time, the write control bits and status bits are shared by both banks:

- Write control bits (PG, FW, EOPIE, WRPERRIE, PGSERRIE, STRBERRIE, INCERRIE, and all related clear bits) control Bank1 and Bank2 simultaneously.
- Write status flags report (WBNE, DBNE, EOP, WRPERR, PGSERR, STRBERR, INCERR, BSY) errors for Bank1 and Bank2 simultaneously.

7.9.2 Nonsecure write protection error (WRPERR)

When an illegal, nonsecure erase/program operation is attempted in the nonvolatile memory, the flash memory interface sets the write protection error flag WRPERR in the FLASH_SR register.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A page that is write-protected with WRPGn
- An HDP area while the HDPL has made it inaccessible
- Secure (TZ) state not matching the erased memory attribute
- Attempt to erase a privilege page within unprivileged mode

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- System flash memory
- A user page that is write-protected with WRPGn
- An OTP block, locked with LOCKBL
- A read-only section
- A reserved area
- Secure area
- Privileged area from within the unprivileged mode
- OBK access conditions check error

When the WRPERR flag is raised, the operation is rejected and nothing is changed in the corresponding bank. If this error is detected, the write buffer is invalidated.

Note: The WRPERR flag must be cleared before any erase/program operation.

The WRPERR flag is cleared by setting the WRPERR bit to 1 in the FLASH_CCR register.

If the WRPERRIE bit in the FLASH_CR register is set to 1, an interrupt is generated when the WRPERR flag is raised (see *Section 7.10* for details).

7.9.3 Secure write protection error (WRPERR)

When an illegal secure erase/program operation is attempted in the nonvolatile memory, the flash memory interface sets the write protection error flag WRPERR in the FLASH_SSR register.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A page that is write-protected with WRPGn
- An HDP area while the HDPL has made it inaccessible
- Secure (TZ) state not matching the erased memory attribute
- Attempt to erase a privileged page within unprivileged mode

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- System flash memory
- A user page that is write-protected with WRPGn
- An OTP block, locked with LOCKBL
- A read-only section
- A reserved area
- Nonsecure area
- Privileged area from within unprivileged mode
- OBK access condition check error

When the WRPERR flag is raised, the operation is rejected and nothing is changed in the corresponding bank.

If this error is detected the write buffer is invalidated.

Note: The WRPERR flag must be cleared before any erase/program operation.

The WRPERR flag is cleared by setting WRPERR bit to 1 in FLASH_SCCR register.

If the WRPERRIE bit in the FLASH_SCR register is set to 1, an interrupt is generated when the WRPERR flag is raised (see *Section 7.10* for details).

7.9.4 Nonsecure programming sequence error (PGSERR)

When the nonsecure programming sequence is incorrect, the flash memory interface sets the programming sequence error flag PGSERR in the FLASH_SR register.

More specifically, the PGSERR flag is set when one of the conditions below is met:

- An error like INCERR, WRPERR, PGSERR, STRBERR, OPTCHANGEERR, OBKERR, or OBKWERR has not been cleared before requesting a new write or erase operation, OBK operation, or options change.
- For erase, PGSERR is set if:
  - Missing erase operation (FLASH_CR): STRT = 1 with (MER = 0, PER = 0 and BER = 0)
  - Page and bank erase requested at the same time (NSSTRT and more than one of MER, PER and BER)
  - PG set during erase operation (it ensures that no erase req and write req happen at the same time): (STRT = 1) with (PG = 1)
  - Erase operation is started while write buffer is waiting for next data: (STRT = 1) with WBNE = 1.
  - STRT = 1 set by secure access.
  - Erase operation is started while DBNE = 1.

- For programming, PGSERR is set if:
  - Missing write flag: a write operation is requested but the program enable bit PG has not been set in FLASH_CR register prior to the request.
  - Write operation to flash high-cycle data is requested but PG = 0.
  - AHB write request is received and (PER = 1, BER = 1 or MER = 1)
  - 16 bit data access requested while WBNE = 1
- For options change, PGSERR is set if:
  - Option change is started while write buffer is waiting for next data: OPTSTRT = 1 with WBNE = 1
  - OPTSTRT set with nonsecure DBNE = 1.
- For option byte key storage access, PGSERR is set if:
  - Error flags from previous operation not cleared
  - ALT_SECT_ERASE and SWAP_SECT_REQ in FLASH_OBKCFGR are set at the same time
  - SWAP_OFFSET value is wrong: when SWAP_SECT_REQ is set to 1, the SWAP_OFFSET should be equal or greater than OBK_HDPL< N - 1 > _SWAP_OFFSET
  - ALT_SECT_ERASE or OBK_SWAP set while DBNE = 1
  - OBK-HDPL is incorrect and nonsecure OBK_SWAP is requested
  - If OBKSWAP or OBK alternate erase is started while write buffer is waiting for new data (WBNE = 1)
- For PUF operation, the NSPGSERR is set if:
  - NS_PUF_LAUNCH is set and FLASH_SR flags are not cleare.
  - NS_PUF_LAUNCH is set with NSPG = 1
  - NS_PUF_LAUNCH is set with NSSTRT = 1
  - NS_PUF_LAUNCH is set with NSWBNE = 1
  - NS_PUF_LAUNCH is set with NSDBNE = 1

When the PGSERR flag is raised as a consequence of a failed write attempt (flash programming), the current program operation is aborted and nothing is changed in the corresponding bank. The write data buffer is also invalidated.

Note: When the PGSERR flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above-mentioned protection mechanism. It is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation was successful by reading back the value in the flash memory.

The PGSERR flag also blocks any new program operation. This means that PGSERR must be cleared before starting a new program operation.

The PGSERR flag is cleared by PGSERR bit to 1 in FLASH_CCR register.

If the PGSERRIE bit in the FLASH_CR register is set to 1, an interrupt is generated when PGSERR flag is raised. See *Section 7.10* for details.

7.9.5 Secure programming sequence error (PGSERR)

When the secure programming sequence is incorrect, the flash interface sets the programming sequence error flag PGSERR in the FLASH_SSR register.

More specifically, the PGSERR flag is set when one of the conditions below is met:

- An error like INCERR, WRPERR, PGSERR, STRBERR, OBKERR, or OBKWERR has not been cleared before requesting a new write or erase operation, OBK operation or options change.
- For erase, PGSERR is set if:
  - Missing erase operation (FLASH_SCR): STRT = 1 with (PER = 0 and BER = 0, MER = 0)
  - Page and bank erase requested at the same time (SECSTRT and more than one of MER, PER, and BER)
  - PG set during erase operation (it ensures that no erase request and write request happen at the same time): (STRT = 1) with (PG = 1)
  - Erase operation is started while write buffer is waiting for next data: (STRT = 1) with WBNE = 1.
  - Operation started while secure DBNE = 1.
- For programming, PGSERR is set if:
  - Missing write flag: a write operation is requested but the program enable bit PG has not been set in FLASH_SCR register prior to the request
  - Write operation to flash high-cycle data is requested but PG = 0
  - AHB write request is received and (PER = 1, SBER = 1, or MER = 1)
  - 16-bit data access requested while WBNE = 1
- For options change, PGSERR is set if:
  - Option change is started while write buffer is waiting for next data: OPTSTRT = 1 with WBNE = 1
  - OPTSTRT set with secure DBNE = 1
- For options byte key storage access, SECPGSERR is set if:
  - Error flags from previous operation not cleared
  - ALT_SECT_ERASE and SWAP_SECT_REQ are set at the same time
  - SWAP_OFFSET value is wrong: when SWAP_SECT_REQ = 1, SWAP_OFFSET must be equal or greater than OBK_HDPL<N-1>_SWAP_OFFSET.
  - SWAP_SECT_REQ or OBK_SWAP set while DBNE = 1
  - OBK-HDPL is incorrect and secure OBK_SWAP is requested
  - If OBKSWAP or OBK alternate erase is started while write buffer is waiting for new data (WBNE = 1)
- For PUF operation, the SECPGSERR is set if:
  - SEC_PUF_LAUNCH is set and FLASH_SSR flags are not cleared
  - SEC_PUF_LAUNCH is set with SECPG = 1
  - SEC_PUF_LAUNCH is set with SECSTRT = 1
  - SEC_PUF_LAUNCH is set with SECWBNE = 1
  - SEC_PUF_LAUNCH is set with SECDBNE = 1

When the PGSERR flag is raised as a consequence of a failed write attempt (flash programming), the current program operation is aborted and nothing is changed in the corresponding bank. The write data buffer is also invalidated.

Note: When the PGSERR flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. It is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation has been successful by reading back the value in the flash memory.

The PGSERR flag also blocks any new program operation. This means that PGSERR must be cleared before starting a new program operation.

The PGSERR flag is cleared by setting SECPGSERR bit to 1 in FLASH_SCCR register.

If the PGSERRIE bit in the FLASH_SCR register is set to 1, an interrupt is generated when the PGSERR flag is raised. See *Section 7.10* for details.

7.9.6 Nonsecure strobe error (STRBERR)

When the nonsecure application software writes several times to the same byte in the write buffer, the flash memory interface sets the strobe error flag STRBERR (FLASH_SR) whatever the target bank of the write access.

When the STRBERR flag is raised, the current program operation is aborted and the write buffer is invalidated.

The STRBERR flag is cleared by setting STRBERR bit to 1 in FLASH_CCR register.

If the STRBERRIE bit in the FLASH_CR register is set to 1, an interrupt is generated when the STRBERR flag is raised. See *Section 7.10* for details.

7.9.7 Secure strobe error (STRBERR)

When the secure application software writes several times to the same byte in the write buffer, the memory sets the strobe error flag STRBERR (FLASH_SSR) whatever the target bank of the write access.

When the STRBERR flag is raised, the current program operation is aborted and the write buffer is invalidated.

The STRBERR flag is cleared by setting the STRBERR bit to 1 in the FLASH_SCCR register.

If the STRBERRIE bit in the FLASH_SCR register is set to 1, an interrupt is generated when the STRBERR flag is raised. See *Section 7.10* for details.

7.9.8 Nonsecure inconsistency error (INCERR)

When a programming inconsistency in a nonsecure access operation is detected, the flash memory interface sets the inconsistency error flag INCERR in the FLASH_SR register.

More specifically, the INCERR flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One AHB master starts a write operation, but before the buffer is full, another master starts a new write operation at the same address or at a different address.
  - ALT_SECT in FLASH_OBKCFGR changed while filling the write buffer.

Note: The INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

It is recommended to follow this sequence to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when the INCERR flag is raised.
2. Stop all write requests to the memory.
3. Clear the INCERR bit.
4. Restart the write operations from where they have been interrupted.

The INCERR flag is cleared by setting the INCERR bit to 1 in the FLASH_CCR register.

If the INCERRIE bit in the FLASH_CR register is set to 1, an interrupt is generated when INCERR flag is raised (see *Section 7.10* for details).

7.9.9 Secure inconsistency error (INCERR)

When a programming inconsistency is detected, the flash interface sets the inconsistency error flag INCERR in register FLASH_SSR.

More specifically, the INCERR flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One AHB master starts a write operation, but before the buffer is full, another master starts a new write operation at the same address or at a different address.

Note: The INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

It is recommended to follow the sequence below to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when the INCERR flag is raised.
2. Stop all write requests to the flash memory.
3. Clear the INCERR bit.
4. Restart the write operations from where they have been interrupted.

The INCERR flag is cleared by setting the INCERR bit to 1 in the FLASH_SCCR register.

If the INCERRIE bit in the FLASH_SCR register is set to 1, an interrupt is generated when the INCERR flag is raised (see *Section 7.10* for details).

7.9.10 Error correction code error (ECCC, ECCD)

When a single-error correction is detected during a read, the flash memory interface sets the single-error correction flag ECCC in the FLASH_ECCCORR register.

When two ECC errors are detected during a read, the flash memory interface sets the double error detection flag ECCD in the FLASH_ECCDETR register.

When the ECCC flag is raised, the corrected read data are returned. The application can ignore the error and request new read operations. When the ECCD the flag is raised, an NMI is generated, which can be masked in SBS registers (*SBS ECC NMI mask register (SBS_ECCNMIR)*) for data access (OTP, data area, RO data). Software must invalidate the instruction cache (CACHEINV = 1) in the NMI interrupt service routine when the ECCD flag is set.

When the ECCC or ECCD flag is raised, the address of the flash word that generated the error is saved in the FLASH_ECCCORR (FLASH_ECCDETR) register. If the address corresponds to a read-only area or to an OTP area or flash high-cycle data, the OTP_ECC bit is also set to 1 in the FLASH_ECCCORRR (FLASH_ECCDETR) register. This register is automatically cleared when the associated flag that generated the error is reset.

A BK_ECC flag indicates in which flash bank the error occurred.

A SYSF_ECC flag indicates an error detected in the system flash area.

Table 78. Locating ECC failure

OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Flash area	ADDR_ECC min	ADDR_ECC max
0	0	0/1	0	0	User flash	0x0000	0xFFFF
0	1	0/1	0	0	System flash	0x0000	0x0FFF
1	0	0	0	0	OTP	0x0600	0x07FF
0	0	1	0	1	OBKeys	0x0000	0x03FF
0	0	0/1	1	0	Data area page 7	0xX200	0xX3FF
0	0	0/1	1	0	Data area page 6	0xX400	0xX5FF
0	0	0/1	1	0	Data area page 5	0xX600	0xX7FF
0	0	0/1	1	0	Data area page 4	0xX800	0xX9FF
0	0	0/1	1	0	Data area page 3	0xXA00	0xXBFF
0	0	0/1	1	0	Data area page 2	0xXC00	0xXDFF
0	0	0/1	1	0	Data area page 1	0xX000	0xX1FF
0	0	0/1	1	0	Data area page 0	0xXE00	0xXFFF

Note: In case of successive single correction or double detection errors, only the address corresponding to the first error is stored in the FLASH_ECCCORR (FLASH_ECCDETR) *register.*

It is mandatory to clear the ECCC or ECCD flags before starting a new read operation.

As the ECC interface is shared by the two banks, if the same error is registered simultaneously, a physical Bank1 error is registered. Both errors are registered if one bank reports ECCD and the other bank ECCC.

ECCC (respectively ECCD) flag is cleared by setting to 1 the ECCC bit (respectively ECCD bit) in the FLASH_ECCCORR (FLASH_ECCDETR) register.

If the ECCC bit in the FLASH_ECCCORR register is set to 1, an interrupt is generated when the ECCC flag is raised. Only NMI is generated for the ECCD. See *Section 7.10* for details.

7.9.11 Illegal access (ILAFM/ILAP)

Illegal access is a signal to the TZIC, triggering a secure interrupt in there. It is complementary to the other interrupts and only generated when the TZ is enabled (TZ_STATE = 0xB4).

It can be masked only on GTZC level.

ILAFM is generated when rules on the secure flash memory access are violated:

- Attempt to access a secure memory location in nonsecure mode.
- Attempt to access a nonsecure memory location in secure mode.

ILAP is generated on:

- Attempt to access a secure register in nonsecure mode.

If ILAFM is detected during write, the write buffer is invalidated.

Note: ILAFM and ILAP has no flag within the flash memory controller to clear to resume operation. It is dealt within the TZIC.

7.9.12 Option-byte change error (OPTCHANGEERR)

When the flash memory interface finds an error during an option change operation, it aborts the operation and sets the option byte change error flag OPTCHANGEERR in the FLASH_SR register.

The error is raised after the OPTSTRT bit set if:

- TZEN is not set and transition to TZ-Closed or NS-Regression state is requested. Also any other forbidden PRODUCT_STATE transition (*Table 68*).
- SWAP_BANK is locked out by SWAP_LOCK.
- OB modification is attempted in wrong HDPL (selected PRODUCT_STATE transitions)
- Attempt to modify OB with invalid value:
  - New EPOCH value not greater than current
  - Value not recognized by the specification (enumerated magic numbers only)
- Attempt to modify OB in product state where it is not allowed (usually a state open for debug is required for change).

Note: Exceptions and details provided in the OB description (see Section 7.4.8 and Table 48).

The OPTCHANGEERR flag is cleared by setting the OPTCHANGEERR bit to 1 in the FLASH_CCR register.

If the OPTCHANGEERRIE bit in the FLASH_CR register is set to 1, an interrupt is generated when the OPTCHANGEERR flag is raised (see *Section 7.10* for details).

It is mandatory to clear the OPTCHANGEERR flag before starting a new operation (option change, erase, or write).

7.9.13 Miscellaneous HardFault errors

The following events generate a bus error on the corresponding bus interface:

- On the main AHB system bus for access targeting code and data with 9 bits ECC.
  - Fetching from the secure user flash memory in nonsecure mode.
  - Fetching from the nonsecure user flash memory in secure mode.
  - Access to an invalid address (including data addresses forbidden for code use).
  - Fetching from the HDP area with incorrect HDPL value.
  - Fetching from a privilege area in unprivileged mode.
- On the AHB configuration or system bus for accesses targeting OTP/RO (all addresses using 6 bits ECC):
  - wrong key input to FLASH_KEYR/SKEYR or FLASH_OPTKEYR.
  - 8-bit access to system AHB interface.
  - Wrong unlock sequence on a register.

7.9.14 OBK error cases (OBKERR, OBKWERR)

Four error types can be detected while filling the OBK write buffer: STRBERR, INCERR, WRPERR, or OBKERR. The OBKERR is specific to the OBK access and cannot be raised by access to any other flash memory asset.

Note: For STRBERR, INCERR, and WRPERR, nonsecure or secure flags are used depending on the AHB access type (secure or nonsecure). If a nonsecure access request is received, the nonsecure error flags are used. If a secure access request is received, the secure error flags are used.

For OBKERR and OBKWERR, nonsecure or secure flags are used depending on the value of the secure state. If the SOC is in secure state (TZ_STATE active), the secure error flags (OBKERR and OBKWERR) are used. Otherwise the nonsecure error flags (OBKERR and OBKWERR) are used.

OBKERR is raised when the OBK-HDPL (input signal from SBS) does not match the HDPL associated with the key during read or write access, and the TZ and PRIV attributes are correct (See *Section 7.4.2: Option byte loading* for more details).

If OBKERR is raised during write access, the write buffer is flushed. If OBKERR is raised on read access, the write buffer remains intact. In both cases, the flag must be cleared to enable write or erase access to the flash memory.

OBKWERR is detected after filling the write buffer and raised if:

- The address is not virgin during a write access.
- The OBK selector in the alternate page is not virgin during a swap operation.

If OBKWERR is raised, the write buffer is flushed. The flag must be cleared to enable write or erase access to the flash memory.

7.10 FLASH interrupts

The flash memory interface can generate a maskable interrupt to signal the following events on a given bank:

- Read and write errors (see *Section 7.9: FLASH error management*):
  - Single ECC error correction during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - Option change operation error
- Security errors (see *Section 7.9: FLASH error management*):
  - Write protection error
  - Illegal access error - signal to TZIC
- Miscellaneous events (described below):
  - End of programming

An NMI is raised on double ECC error detection during a read operation.

Two interrupt lines gather all error sources, one for errors in nonsecure access and another for errors that happen in secure access.

The user can enable or disable the flash interface interrupt sources individually by changing the mask bits in the FLASH_CR/SCR, FLASH_ECCCORR, and FLASH_ECCDETR register. Setting the appropriate mask bit to 1 enables the interrupt.

Note: Prior to writing, the FLASH_CR/SCR register must be unlocked, as explained in Section 7.6.7: Flash memory configuration protection.

Table 79 gives a summary of the available flash memory interface interrupt features. As mentioned in the table below, some flags need to be cleared before triggering a new operation.

Table 79. Flash interrupt request

Interrupt event Event flag	Error flag label in register	Enable control bit	Clear flag to resume operation
End of operation event	EOP	EOPIE	N/A
Write protection error	WRPERR	WRPERRIE	Yes
Programming sequence error	PGSERR	PGSERRIE	Yes
Strobe error	STRBERR	STRBERRIE	Yes
OBK write error	OBKWERR	OBKWERRIE	Yes
OBK general error	OBKERR	OBKERRIE	Yes
Inconsistency error	INCERR	INCERRIE	Yes
Illegal access error	ILAFM	TZEN	No

Table 79. Flash interrupt request (continued)

Interrupt event Event flag	Error flag label in register	Enable control bit	Clear flag to resume operation
Option byte error	OPTCHANGEERR	-	Yes
ECC single error correction event	ECCC	ECCCIE	No
ECC double error detection event(1)	ECCD	Disabled	No

1. NMI.

The status of the individual maskable interrupt sources described in *Table 79* (except for option byte error and ECC) can be read from the FLASH_SR/SSR register. They can be cleared by setting to 1 the adequate bit in the FLASH_CCR/SCCR register.

Note: No unlocking mechanism is required to clear an interrupt.

End of operation event

Setting the end of operation interrupt enable bit (EOPIE) in the FLASH_CR/SCR register enables the generation of an interrupt at the end of an erase operation, a program operation, a PUF operation, or an option byte change.

When managing the OBKey storage area, the EOP is also associated with the OBK swap operation and the alternate area erase operation. The EOP bit in the FLASH_SR/SSR register is also set when one of these events occurs.

Setting the EOP bit in the FLASH_CCR/SCCR register clears the EOP flag.

7.11 FLASH registers

Each register has an assigned offset address and reset value. In case of registers representing option byte value, the reset value is determined by the OBL process. In case of success, the reset value is loaded from OB. In case of OBL failure, a highly restrictive default value is set.

7.11.1 FLASH access control register (FLASH_ACR)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

For more details, refer to *Section 7.3.4* and *Section 7.3.5*.

Address offset: 0x000

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACR_SEC_EN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTEN	Res.	Res.	WRHIGHFREQ[1:0]		LATENCY[3:0]			
							rw			rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ACR_SEC_EN**: This bit restricts the contents of the FLASH_ACR from nonsecure write access.
0: all nonsecure access to FLASH_ACR permitted.
1: nonsecure write access to FLASH_ACR results in illegal access error. The register is read only for nonsecure code.
Note: This bit is only considered when TrustZone is enabled..

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **PRFTEN**: Prefetch enable. When bit value is modified, user must read back ACR register to be sure PRFTEN has been taken into account.
Bits used to control the prefetch.
0: prefetch disabled.
1: prefetch enabled when latency is at least one wait state.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **WRHIGHFREQ[1:0]**: Flash signal delay
These bits are used to control the delay between nonvolatile memory signals during programming operations. Application software has to program them to the correct value depending on the memory interface frequency. Please refer to *Table 39* for details.
Note: No check is performed to verify that the configuration is correct.
Two WRHIGHFREQ values can be selected for some frequencies.

Bits 3:0 **LATENCY[3:0]**: Read latency

These bits are used to control the number of wait states used during read operations on both nonvolatile memory banks. The application software has to program them to the correct value depending on the memory interface frequency and voltage conditions.

0000: zero wait states used to read a word from nonvolatile memory

0001: one wait state used to read a word from nonvolatile memory

0010: two wait states used to read a word from nonvolatile memory

...

0111: seven wait states used to read a word from nonvolatile memory

1111: 15 wait states used to read a word from nonvolatile memory

Note: No check is performed by hardware to verify that the configuration is correct.

7.11.2 FLASH nonsecure key register (FLASH_KEYR)

This register is nonsecure. It can written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_KEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_CR register and allow programming/erasing it. A wrong sequence locks the FLASH_CR register until next system reset.

First key = 0x4567 0123

Second key = 0xCDEF 89AB

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Non-volatile memory nonsecure configuration access unlock key

7.11.3 FLASH secure key register (FLASH_SKEYR)

This register is secure. It can be written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_SKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_SCR register and allow programming/erasing it. A wrong sequence locks the FLASH_SCR register until next system reset.

First key = 0x4567 0123

Second key = 0xCDEF 89AB

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Non-volatile memory secure configuration access unlock key

7.11.4 FLASH option key register (FLASH_OPTKEYR)

This register is nonsecure. It can be written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_OPTKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_OPTCR register. A wrong sequence locks FLASH_OPTCR register until next system reset.

First key = 0x0819 2A3B

Second key = 0x4C5D 6E7F

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: FLASH option bytes control access unlock key

7.11.5 FLASH nonsecure OBK key register (FLASH_OBKKEYR)

This is a write-only register. The following values must be programmed consecutively to unlock FLASH_OBKCFGR register. A wrong sequence locks FLASH_OBKCFGR register until next system reset.

First key = 0x192A 083B

Second key = 0x6E7F 4C5D

This register is nonsecure. This register can be written only by privileged access.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: FLASH nonsecure option bytes keys control access unlock key

7.11.6 FLASH secure OBK key register (FLASH_SOBKKEYR)

This is a write-only register. The following values must be programmed consecutively to unlock FLASH_SOBKCFGR register. A wrong sequence locks FLASH_SOBKCFGR register until next system reset.

First key = 0x192A 083B

Second key = 0x6E7F 4C5D

This register is secure. This register can only be written by privileged access.

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: FLASH secure option bytes keys control access unlock key

7.11.7 FLASH operation status register (FLASH_OPSR)

This register is nonsecure. This register can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x0018

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CODE_OP[2:0]			Res.	Res.	Res.	Res.	OTP_OP	SYSF_OP	BK_OP	DATA_OP	Res.	ADDR_OP[19:16]			
r	r	r					r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_OP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 **CODE_OP[2:0]**: Flash memory operation code
000: No flash operation on going during previous reset
001: Single write operation interrupted
010: OBK alternate page erase
011: Page erase operation interrupted
100: Bank erase operation interrupted
101: Mass erase operation interrupted
110: Option change operation interrupted
111: OBK swap page request

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **OTP_OP**: OTP operation interrupted
Indicates that reset interrupted an ongoing operation in OTP area (or OBKeys area).

Bit 23 **SYSF_OP**: Operation in system flash memory interrupted
Indicates that reset interrupted an ongoing operation in system flash.

Bit 22 **BK_OP**: Interrupted operation bank
It indicates which bank was concerned by operation.

Bit 21 **DATA_OP**: Flash high-cycle data area operation interrupted
It indicates if flash high-cycle data area is concerned by operation.

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **ADDR_OP[19:0]**: Interrupted operation address

7.11.8 FLASH option control register (FLASH_OPTCR)

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait state when no FLASH memory operations are ongoing. The FLASH_OPTCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it while the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x01C

Reset value: 0xX000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	OPTLOCK
														rw	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OPTSTRT**: Option byte start change option configuration bit

OPTSTRT triggers an option byte change operation. The user can set OPTSTRT only when the OPTLOCK bit is cleared to 0. It is set only by Software and cleared when the option byte change is completed or an error occurs (PGSERR or OPTCHANGEERR). It is reseted at the same time as BSY bit.

The user application cannot modify any FLASH_XXX_PRG flash interface register until the option change operation has been completed.

Before setting this bit, the user has to write the required values in the FLASH_XXX_PRG registers. The FLASH_XXX_PRG registers are locked until the option byte change operation has been executed in nonvolatile memory.

Bit 0 **OPTLOCK**: FLASH_OPTCR lock option configuration bit

The OPTLOCK bit locks the FLASH_OPTCR register as well as all _PRG registers. The correct write sequence to FLASH_OPTKEYR register unlocks this bit. If a wrong sequence is executed, or the unlock sequence to FLASH_OPTKEYR is performed twice, this bit remains locked until next system reset.

It is possible to set OPTLOCK by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When OPTLOCK changes from 0 to 1, the others bits of FLASH_OPTCR register do not change.

0: FLASH_OPTCR register unlocked

1: FLASH_OPTCR register locked.

7.11.9 FLASH nonsecure status register (FLASH_SR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x020

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERR	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBNE	PUF_STATE	WBNE	BSY
												r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERR**: Option byte change error flag

OPTCHANGEERR flag indicates that an error occurred during an option byte change operation. When OPTCHANGEERR is set to 1, the option byte change operation did not successfully complete. An interrupt is generated when this flag is raised if the OPTCHANGEERRIE bit of FLASH_CR register is set to 1.

Use FLASH_CCR to clear OPTCHANGEERR.

0: no option byte change errors occurred

1: one or more errors occurred during an option byte change operation.

Note: The OPTSTRT bit in FLASH_OPTCR cannot be set while OPTCHANGEERR is set.

Bit 22 **OBKWERR**: OBK write error flag

OBKWERR flag is raised when the address is not virgin on a write access to the OBK storage. Alternatively also when the OBK selector in the alternate page is not virgin during a swap operation.

0: no OBK write error occurred

1: an OBK write error occurred

Bit 21 **OBKERR**: OBK general error flag

OBKERR flag is raised when the OBK-HDPL signal from the SBS does not match the HDPL value associated with the key slot during access to the key location. Alternatively also when the ALT_SECT is unexpectedly changed while the write buffer is being filled.

0: no OBK general error occurred

1: an OBK general error occurred

Bit 20 **INCERR:** inconsistency error flag

NSINCERR flag is raised when a inconsistency error occurs. An interrupt is generated if INCERRIE is set to 1.

Use FLASH_CCR to clear INCERR.

0: no inconsistency error occurs

1: a inconsistency error occurs

Bit 19 **STRBERR:** strobe error flag

STRBERR flag is raised when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE bit is set to 1. Use FLASH_CCR to clear STRBERR.

0: no strobe error occurred

1: a strobe error occurred

Bit 18 **PGSERR:** programming sequence error flag

PGSERR flag is raised when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set to 1. Use FLASH_CCR to clear PGSERR.

0: no sequence error occurred

1: a sequence error occurred

Bit 17 **WRPERR**: write protection error flag

WRPERR flag is raised when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set to 1. Use FLASH_CCR to clear WRPERR.

0: no write protection error occurred

1: a write protection error occurred

Bit 16 **EOP**: end of operation flag

EOP flag is set when a operation (program/erase) completes. An interrupt is generated if the EOPIE is set to 1. It is not necessary to reset EOP before starting a new operation. EOP bit is cleared in FLASH_CCR register.

0: no operation completed

1: a operation completed

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **DBNE**: data buffer not empty flag

DBNE flag is set when the flash interface is processing 6-bits ECC data in dedicated buffer. This bit cannot be set to 0 by software. The hardware resets it once the buffer is free.

0: data buffer not used

1: data buffer used, wait

Bit 2 **PUF_STATE**: PUF readiness

PUF_STATE flag is set when a PUF operation is completed successfully and PUF is ready for use. The value is not reset by regular reset, only after power up or standby.

0: PUF is not ready. Retry PUF_LAUNCH

1: PUF is ready for use

Bit 1 **WBNE**: write buffer not empty flag

WBNE flag is set when the flash interface is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the events below:

- the application software forces the write operation using FW bit in FLASH_CR
- the memory detects an error that involves data loss

This bit cannot be reset by software writing 0 directly. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: write buffer empty or full

1: write buffer waiting data to complete

Bit 0 **BSY**: busy flag

BSY flag indicates that a flash memory is busy by an operation (write, erase, option byte change, OBK, and PUF operations). It is set at the beginning of a flash memory operation and cleared when the operation finishes, or an error occurs.

0: no programming, erase or option byte change operation being executed

1: programming, erase or option byte change operation being executed

7.11.10 FLASH secure status register (FLASH_SSR)

This register is secure. It can be read only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x024

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBNE	PUF_STATE	WBNE	BSY
												r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **OBKWERR**: OBK write error flag

OBKWERR flag is raised when the address is not virgin on a write access to the OBK storage. Alternatively also when the OBK selector in the alternate page is not virgin during a swap operation.

0: no OBK write error occurred

1: an OBK write error occurred

Bit 21 **OBKERR**: OBK general error flag

OBKERR flag is raised when the OBK-HDPL signal from the SBS does not match the HDPL value associated with the key slot during access to the key location. Alternatively also when the ALT_SECT is unexpectedly changed while the write buffer is being filled.

0: no OBK general error occurred

1: an OBK general error occurred

Bit 20 **INCERR:** inconsistency error flag

INCERR flag is raised when a inconsistency error occurs. An interrupt is generated if INCERRIE is set to 1. Use FLASH_SCCR to clear INCERR.

0: no inconsistency error occurred

1: a inconsistency error occurred

Bit 19 **STRBERR:** strobe error flag

STRBERR flag is raised when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE bit is set to 1. Use FLASH_SCCR to clear STRBERR.

0: no strobe error occurred

1: a strobe error occurred

Bit 18 **PGSERR:** programming sequence error flag

PGSERR flag is raised when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set to 1. Use FLASH_SCCR to clear PGSERR.

0: no sequence error occurred

1: a sequence error occurred

Bit 17 **WRPERR**: write protection error flag

WRPERR flag is raised when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set to 1. Use FLASH_SCCR to clear WRPERR.

0: no write protection error occurred

1: a write protection error occurred

Bit 16 **EOP**: end of operation flag

EOP flag is set when a operation (program/erase) completes. An interrupt is generated if the EOPIE is set to. It is not necessary to reset EOP before starting a new operation. EOP bit is cleared in FLASH_SCCR register.

0: no operation completed

1: a operation completed

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **DBNE**: data buffer not empty flag

DBNE flag is set when the memory interface is processing 6-bits ECC data in dedicated buffer. This bit cannot be set to 0 by software. The hardware resets it once the buffer is free.

0: data buffer not used

1: data buffer used, wait

Bit 2 **PUF_STATE**: PUF readiness

PUF_STATE flag is set when a PUF operation is completed successfully and PUF is ready for use. The value is not reset by regular reset, only after power up or standby.

0: PUF is not ready. Retry PUF_LAUNCH

1: PUF is ready for use.

Bit 1 **WBNE**: write buffer not empty flag

WBNE flag is set when the flash interface is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the events below:

- the application software forces the write operation using FW bit in FLASH_SCR
- the flash interface detects an error that involves data loss

This bit cannot be reset by writing 0 directly by software. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: write buffer empty or full

1: write buffer waiting data to complete

Bit 0 **BSY**: busy flag

BSY flag indicates that a FLASH memory is busy (write, erase, option byte change, OBK operations, PUF operation). It is set at the beginning of a flash memory operation and cleared when the operation finishes or an error occurs.

0: no programming, erase or option byte change operation being executed

1: programming, erase or option byte change operation being executed

7.11.11 FLASH nonsecure control register (FLASH_CR)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait state when no memory operations are ongoing. The FLASH_CR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x028

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERRIE	OBKWERRIE	OBKERRIE	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER	Res.	PUF_LAUNCH	Res.	Res.	PNB[4:0]					STRT	FW	BER	PER	PG	LOCK
rw		rw			rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rs

Bit 31 **BKSEL**: Bank selector bit

Can only be programmed when LOCK is cleared to 0. The bit selects physical bank, SWAP_BANK setting is ignored.

0: Bank1 is selected for Bank erase / page erase / interrupt enable

1: Bank2 is selected for BER / PER

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERRIE**: Option byte change error interrupt enable bit

This bit controls if an interrupt must be generated when an error occurs during an option byte change. It can be programmed only when LOCK bit is cleared to 0.

0: no interrupt is generated when an error occurs during an option byte change

1: an interrupt is generated when and error occurs during an option byte change.

Bit 22 **OBKWERRIE**: OBK write error interrupt enable bit

OBKWERRIE enables generation of interrupt in case of OBK specific write error. This bit can be programmed only when LOCK bit is cleared to 0.

0: no interrupt is generated on OBK write error

1: an interrupt is generated on OBK write error

Bit 21 **OBKERRIE**: OBK general error interrupt enable bit

OBKERRIE enables generating an interrupt in case of OBK specific access error. This bit can be programmed only when LOCK bit is cleared to 0.

0: no interrupt is generated on OBK general access error

1: an interrupt is generated on OBK general access error

Bit 20 **INCERRIE**: inconsistency error interrupt enable bit

When INCERRIE bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation. INCERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a inconsistency error occurs

1: interrupt generated when a inconsistency error occurs.

Bit 19 **STRBERRIE**: strobe error interrupt enable bit

When STRBERRIE bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation. STRBERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a strobe error occurs

1: interrupt generated when strobe error occurs.

Bit 18 **PGSERRIE**: programming sequence error interrupt enable bit

When this bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation. PGSERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a sequence error occurs

1: interrupt generated when sequence error occurs

Bit 17 **WRPERRIE**: write protection error interrupt enable bit

When this bit is set to 1, an interrupt is generated when a protection error occurs during a program operation. WRPERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a protection error occurs

1: interrupt generated when a protection error occurs

Bit 16 **EOPIE**: end of operation interrupt control bit

Setting EOPIE bit to 1 enables the generation of an interrupt at the end of a program or erase operation. EOPIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated at the end of operation.

1: interrupt enabled when at the end of operation

Bit 15 **MER**: Mass erase request

Setting MER bit to 1 requests a mass erase operation (user flash memory only). MER can be programmed only when LOCK is cleared to 0.

If BER or PER are both set, a PGSERR is raised.

0: mass erase not requested

1: mass erase requested

Note: An error is triggered when a mass erase is required and some pages are protected.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **PUF_LAUNCH**: PUF preparation start control bit

PUF_LAUNCH bit is used to launch a PUF operation. PUF_LAUNCH can be set only when LOCK is cleared to 0. The setting is only valid when TZstate = 0xC3 and HDPL is 0 or 1, ignored otherwise.

PUF_LAUNCH is reset at the end of the operation or when an error occurs. It cannot be reset by software.

Bits 12:11 Reserved, must be kept at reset value.

Bits 10:6 **PNB[4:0]**: page erase selection number

These bits are used to select the target page for an erase operation (they are unused otherwise). They can be programmed only when LOCK is cleared to 0.

0x00: Page 0 selected

0x01: Page 1 selected

..

0x1F: Page 31 selected

Note: For products with lower page count consider the extra PNB bits reserved.

Bit 5 **STRT**: erase start control bit

STRT bit is used to start a page erase or a bank erase operation. STRT can be programmed only when LOCK is cleared to 0.

STRT is reset at the end of the operation or when an error occurs. It cannot be reseted by software.

Bit 4 **FW:** write forcing control bit

FW forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW can be programmed only when LOCK is cleared to 0.

The memory resets FW when the corresponding operation has been acknowledged.

Note: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it leads to permanent ECC error.

Write forcing is effective only if the write buffer is not empty and was filled by nonsecure access (in particular, FW does not start several write operations when the force-write operations are performed consecutively).

Since there is just one write buffer, FW can force a write in Bank1 or Bank2.

Bit 3 **BER**: erase request

Setting BER bit to 1 requests a bank erase operation (user flash memory only). BER can be programmed only when LOCK is cleared to 0.

If MER and PER are also set, a PGSERR is raised.

0: bank erase not requested

1: bank erase requested

Note: Write protection error is triggered when a bank erase is required and some pages are protected.

Bit 2 **PER**: page erase request

Setting PER bit to 1 requests a page erase. PER can be programmed only when LOCK is cleared to 0.

If MER and PER are also set, a PGSERR is raised.

0: page erase not requested

1: page erase requested

Bit 1 **PG**: programming control bit
PG can be programmed only when LOCK is cleared to 0.
PG allows programming in Bank1 and Bank2.
0: programming disabled
1: programming enabled

Bit 0 **LOCK**: configuration lock bit
This bit locks the FLASH_CR register. The correct write sequence to FLASH_KEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_KEYR is performed twice, this bit remains locked until the next system reset.
LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_CR register do not change.
0: FLASH_CR register unlocked
1: FLASH_CR register locked

7.11.12 FLASH secure control register (FLASH_SCR)

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait state when no memory operations are ongoing. The FLASH_SCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x02C

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKSEL	Res.	INV	Res.	Res.	Res.	Res.	Res.	Res.	OBKWERRIE	OBKERRIE	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE
rw		rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER	Res.	PUF_LAUNCH	Res.	Res.	PNB[4:0]					STRT	FW	BER	PER	PG	LOCK
rw		rw			rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rs

Bit 31 **BKSEL**: bank selector bit
Can only be programmed when LOCK is cleared to 0. The bit selects physical bank, SWAP_BANK setting is ignored.
0: Bank1 is selected for Bank erase / page erase / interrupt enable
1: Bank2 is selected for BER / PER

Bit 30 Reserved, must be kept at reset value.

Bit 29 **INV**: Flash memory security state invert.
This bit inverts the flash memory security state.

Bits 28:23 Reserved, must be kept at reset value.

Bit 22 **OBKWERRIE**: OBK write error interrupt enable bit

OBKWERRIE enables generation of interrupt in case of OBK specific write error. OBKWERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt is generated on OBK write error

1: an interrupt is generated on OBK write error

Bit 21 **OBKERRIE**: OBK general error interrupt enable bit

OBKERRIE enables generating an interrupt in case of OBK specific access error. OBKERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt is generated on OBK general access error

1: an interrupt is generated on OBK general access error

Bit 20 **INCERRIE**: inconsistency error interrupt enable bit

When INCERRIE bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation. INCERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a inconsistency error occurs

1: interrupt generated when a inconsistency error occurs.

Bit 19 **STRBERRIE**: strobe error interrupt enable bit

When STRBERRIE bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation. STRBERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a strobe error occurs

1: interrupt generated when strobe error occurs.

Bit 18 **PGSERRIE**: programming sequence error interrupt enable bit

When PGSERRIE bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation. PGSERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a sequence error occurs

1: interrupt generated when sequence error occurs

Bit 17 **WRPERRIE**: write protection error interrupt enable bit

When WRPERRIE bit is set to 1, an interrupt is generated when a protection error occurs during a program operation. WRPERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a protection error occurs

1: interrupt generated when a protection error occurs

Bit 16 **EOPIE**: end of operation interrupt control bit

Setting EOPIE bit to 1 enables the generation of an interrupt at the end of a program/erase operation. EOPIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated at the end of operation.

1: interrupt enabled when at the end of operation

Bit 15 **MER**: mass erase request

Setting MER bit to 1 requests a mass erase operation (user flash memory only). MER can be programmed only when LOCK is cleared to 0.

If BER or PER are also set, a PGSERR is raised.

0: mass erase not requested

1: mass erase requested

Note: An error is triggered when a mass erase is required and some pages are protected.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **PUF_LAUNCH**: PUF preparation start control bit

PUF_LAUNCH bit is used to launch a PUF operation. PUF_LAUNCH can be set only when LOCK is cleared to 0 and HDPL is 0 or 1.

PUF_LAUNCH is reseted at the end of the operation or when an error occurs. It cannot be reset by software.

Bits 12:11 Reserved, must be kept at reset value.

Bits 10:6 **PNB[4:0]**: page erase selection number

These bits are used to select the target page for an erase operation (they are unused otherwise). They can be programmed only when LOCK is cleared to 0.

0x00: Page 0 selected

0x01: Page 1 selected

..

0x1F: Page 31 selected

Note: For products with lower page count consider the extra PNB bits reserved.

Bit 5 **STRT**: erase start control bit

STRT bit is used to start a page erase or a bank erase operation. STRT can be programmed only when LOCK is cleared to 0.

STRT is reseted at the end of the operation or when an error occurs. It cannot be reset by software.

Bit 4 **FW:** write forcing control bit

FW forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW can be programmed only when LOCK is cleared to 0.

The memory resets FW when the corresponding operation has been acknowledged.

Note: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it leads to permanent ECC error.

Write forcing is effective only if the write buffer is not empty and was filled by secure access (in particular, FW does not start several write operations when the force-write operations are performed consecutively).

Since there is just one write buffer, FW can force a write in Bank1 or Bank2.

Bit 3 **BER**: erase request

Setting BER bit to 1 requests a bank erase operation (user flash memory only). BER can be programmed only when LOCK is cleared to 0.

If MER and PER are also set, a PGSERR is raised.

0: bank erase not requested

1: bank erase requested

Note: Write protection error is triggered when a bank erase is required and some pages are protected.

Bit 2 **PER**: page erase request

Setting PER bit to 1 requests a page erase. PER can be programmed only when LOCK is cleared to 0.

If BER and MER are also set, a PGSERR is raised.

0: page erase not requested

1: page erase requested

Bit 1 **PG**: programming control bit

PG can be programmed only when LOCK is cleared to 0.

PG allows programming in Bank1 and Bank2.

0: programming disabled

1: programming enabled

Bit 0 **LOCK**: configuration lock bit

This bit locks the FLASH_SCR register. The correct write sequence to FLASH_SKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_SKEYR is performed twice, this bit remains locked until the next system reset.

LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_SCR register do not change.

0: FLASH_SCR register unlocked

1: FLASH_SCR register locked

7.11.13 FLASH nonsecure clear control register (FLASH_CCR)

This register is nonsecure. It can be written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERR	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP
								w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERR:** Clear the flag corresponding flag in FLASH_SR by writing this bit.

Bit 22 **OBKWERR**: OBKWERR flag clear bit.

Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_SR register.

Bit 21 **OBKERR**: OBKERR flag clear bit.

Setting this bit to 1 resets to 0 OBKERR flag in FLASH_SR register.

Bit 20 **INCERR**: INCERR flag clear bit

Setting this bit to 1 resets to 0 INCERR flag in FLASH_SR register.

Bit 19 **STRBERR**: STRBERR flag clear bit

Setting this bit to 1 resets to 0 STRBERR flag in FLASH_SR register.

Bit 18 **PGSERR**: PGSERR flag clear bit
Setting this bit to 1 resets to 0 PGSERR flag in FLASH_SR register.

Bit 17 **WRPERR**: WRPERR flag clear bit
Setting this bit to 1 resets to 0 WRPERR flag in FLASH_SR register.

Bit 16 **EOP**: EOP flag clear bit
Setting this bit to 1 resets to 0 EOP flag in FLASH_SR register.

Bits 15:0 Reserved, must be kept at reset value.

7.11.14 FLASH secure clear control register (FLASH_SCCR)

This register is secure. It can be written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP
									w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **OBKWERR**: OBKWERR flag clear bit
Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_SSR register.

Bit 21 **OBKERR**: OBKWERR flag clear bit
Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_SSR register.

Bit 20 **INCERR**: INCERR flag clear bit
Setting this bit to 1 resets to 0 INCERR flag in FLASH_SSR register.

Bit 19 **STRBERR**: STRBERR flag clear bit
Setting this bit to 1 resets to 0 STRBERR flag in FLASH_SSR register.

Bit 18 **PGSERR**: PGSERR flag clear bit
Setting this bit to 1 resets to 0 PGSERR flag in FLASH_SSR register.

Bit 17 **WRPERR**: WRPERR flag clear bit
Setting this bit to 1 resets to 0 WRPERR flag in FLASH_SSR register.

Bit 16 **EOP**: EOP flag clear bit
Setting this bit to 1 resets to 0 EOP flag in FLASH_SSR register.

Bits 15:0 Reserved, must be kept at reset value.

7.11.15 FLASH privilege configuration register (FLASH_PRIVCFGR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register can be read by both privileged and unprivileged access. PRIV is a nonsecure bit. SPRIV is a secure bit. It can be written only by privileged mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SPRIV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PRIV**: privilege attribute for non secure registers
- 0: access to non secure registers is always granted
- 1: access to non secure registers is denied in case of unprivileged access.

Bit 0 **SPRIV**: privilege attribute for secure registers
- 0: access to secure registers is always granted
- 1: access to secure registers is denied in case of unprivileged access.

7.11.16 FLASH nonsecure OBK configuration register (FLASH_OBKCFGR)

Register is only available when TZ_STATE = 0xC3. This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can only be accessed by privilege code.

The FLASH_OBKCFGR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x040

Reset value: 0x01FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_OFFSET[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALT_SECT_ERASE	ALT_SECT	SWAP_SECT_REQ	LOCK
												rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **SWAP_OFFSET[8:0]**: key index (offset /16 bytes) pointing for next swap.
0x00: means that no OBK is copied from current to alternate OBK page during SWAP operation.
0x01 means that only the first OBK data (128 bits) are copied from current to alternate OBK page
0x02 means that the two first OBK data are copied …
…
0x1FF: means that all OBK data (511) are copied

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **ALT_SECT_ERASE**: alternate page erase bit
0: do not touch OBK page
1: erase the alternate OBK page
When ALT_SECT bit is set, use this bit to generate an erase command for the OBK alternate page. It is set only by Software and cleared when the OBK swap operation is completed or an error occurs (PGSERR). It is reseted at the same time as BUSY bit.

Bit 2 **ALT_SECT**: alternate page bit
0: current OBK page is mapped to OBK address range for access
1: alternate OBK page is mapped to OBK address range for access
This bit must not change while filling the write buffer, otherwise an error (OBKERR) is generated

Bit 1 **SWAP_SECT_REQ**: OBK swap page request bit
0: no swap requested
1: launch the page swap
When set, all the OBKs not updated in the alternate page are copied from current page to alternate one.
The SWAP_OFFSET value must have a minimum value for the swap to be launched. Minimum value is 16 for OBK-HDPL = 1, 144 for OBK-HDPL = 2 and 256 for OBK-HDPL = 3.

Bit 0 **LOCK**: OBKCFGR lock option configuration bit
This bit locks the register. The correct write sequence to FLASH_OBKKEYR register unlocks it. If a wrong sequence is executed, or if the unlock sequence is performed twice, this bit remains locked until the next system reset. LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_CR register do not change.
0: FLASH_OBKCFGR register unlocked
1: FLASH_OBKCFGR register locked

7.11.17 FLASH secure OBK configuration register (FLASH_SOBKCFGR)

This register can be accessed only in secure privileged mode. The FLASH_SOBKCFGR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x044

Reset value: 0x01FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_OFFSET[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALT_SECT_ERASE	ALT_SECT	SWAP_SECT_REQ	LOCK
												rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **SWAP_OFFSET[8:0]**: key index (offset /16 bits) pointing for next swap.
0x0: no OBK is copied from current to alternate OBK page during SWAP operation.
0x1: only the first OBK data (128 bits) are copied from current to alternate OBK page
0x2: the two first OBK data are copied …
…
0x1FF: that the 511 first OBK data are copied

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **ALT_SECT_ERASE**: alternate page erase bit
0: do not touch OBK page
1: erase the alternate OBK page
When ALT_SECT bit is set, use this bit to generate an erase command for the OBK alternate page. It is set only by Software and cleared when the OBK swap operation is completed or an error occurs (PGSERR). It is reset at the same time as the BUSY bit.

Bit 2 **ALT_SECT**: alternate page bit
0: current OBK page is mapped to OBK address range for access
1: alternate OBK page is mapped to OBK address range for access
This bit must not change while filling the write buffer, otherwise an error is generated

Bit 1 **SWAP_SECT_REQ**: OBK swap page request bit
0: no swap requested
1: launch the page swap
When set, all the OBKs not updated in the alternate page are copied from current page to alternate one.
The SWAP_OFFSET value must have a minimum value for the swap to be launched.
Minimum value is 16 for OBK-HDPL = 1, 144 for OBK-HDPL = 2 and 256 for OBK-HDPL = 3.

Bit 0 **LOCK**: OBKCFGR lock option configuration bit

This bit locks the register. The correct write sequence to FLASH_SOBKKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence is performed twice, this bit remains locked until the next system reset. LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_CR register do not change.

0: FLASH_OBKCFGR register unlocked
1: FLASH_OBKCFGR register locked

7.11.18 FLASH HDP extension register (FLASH_HDPEXTR)

All bits in this register are nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

The register can only be modified in HDPL<=2.

The values in this register cannot be decremented. Attempt to write a lower value that current is ignored.

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_EXT[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_EXT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **HDP2_EXT[4:0]**: HDP area extension in 8 Kbytes pages in Bank2. Extension is added after the END page (included).

Note: For products with lower page count consider the extra bits 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **HDP1_EXT[4:0]**: HDP area extension in 8 Kbytes pages in Bank1. Extension is added after the END page (included).

Note: For products with lower page count consider the extra bits 4 or 3 reserved.

7.11.19 FLASH option status register (FLASH_OPTSR_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x2D30 EDD8

Default value: 0x2D30 5CD8 (value in case of double ECC issue during OBL)

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x050

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BOOT_UBE[7:0]								IWDG_ STDBY	IWDG_ STOP	Res.	Res.	Res.	Res.
		r	r	r	r	r	r	r	r	r	r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_STATE[7:0]								NRST_ STDBY	NRST_ STOP	Res.	WWDG _SW	IWDG_ SW	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r		r	r			

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:22 **BOOT_UBE[7:0]:** Available only on cryptography enabled devices.
Unique boot entry control, selects either ST or OEM iRoT for secure boot.
0xC3: ST-iRoT (system flash) selected
0xB4: OEM-iRoT (user flash) selected. In Open PRODUCT_STATE this value selects bootloader. Default value.

Bit 21 **IWDG_STDBY**: IWDG Standby mode freeze option status bit
When set the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog 1 frozen in Standby mode
1: Independent watchdog 1 keep running in Standby mode.

Bit 20 **IWDG_STOP**: IWDG Stop mode freeze option status bit
When set the independent watchdog IWDG is in system Stop mode.
0: Independent watchdog 1 frozen in system Stop mode
1: Independent watchdog 1 keep running in system Stop mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:8 **PRODUCT_STATE[7:0]**: Life state code (based on Hamming 8,4). See in *Section 7.6.11: Product state transitions*.

Bit 7 **NRST_STDBY**: Core domain Standby entry reset option status bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain Stop entry reset option status bit
0: a reset is generated when entering Stop mode on core domain
1: no reset generated when entering Stop mode on core domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WWDG_SW**: WWDG control mode option status bit
0: WWDG watchdog is controlled by hardware
1: WWDG watchdog is controlled by software

Bit 3 **IWDG_SW**: IWDG control mode option status bit
0: IWDG watchdog is controlled by hardware
1: IWDG watchdog is controlled by software

Bits 2:0 Reserved, must be kept at reset value.

7.11.20 FLASH option status register (FLASH_OPTSR_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

Address offset: 0x054

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BOOT_UBE[7:0]								IWDG_STDBY	IWDG_STOP	Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw			

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:22 **BOOT_UBE[7:0]:** Available only on cryptography enabled devices.
Unique boot entry control, selects either ST or OEM iRoT for secure boot.
0xC3: ST-iRoT (system flash) selected
0xB4: OEM-iRoT (user flash) selected.
In Open PRODUCT_STATE this value selects bootloader. Default value.

Bit 21 **IWDG_STDBY**: IWDG Standby mode freeze option status bit
When set the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode.

Bit 20 **IWDG_STOP**: IWDG Stop mode freeze option status bit
When set the independent watchdog IWDG is in system Stop mode.
0: Independent watchdog frozen in system Stop mode
1: Independent watchdog keep running in system Stop mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:8 **PRODUCT_STATE[7:0]**: Life state code (based on Hamming 8,4). See in *Section 7.6.11: Product state transitions*.

Bit 7 **NRST_STDBY**: Core domain Standby entry reset option configuration bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain Stop entry reset option configuration bit
0: a reset is generated when entering Stop mode on core domain
1: no reset generated when entering Stop mode on core domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WWDG_SW**: WWDG control mode option configuration bit
0: WWDG watchdog is controlled by hardware
1: WWDG watchdog is controlled by software

Bit 3 **IWDG_SW**: IWDG control mode option configuration bit
0: IWDG watchdog is controlled by hardware
1: IWDG watchdog is controlled by software

Bits 2:0 Reserved, must be kept at reset value.

7.11.21 FLASH nonsecure EPOCH register (FLASH_EPOCHR_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x0000 0001

Default value: 0x00FF FFFF (value in case of double ECC issue during OBL)

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x060

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPOCH[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **EPOCH[23:0]:** Non-volatile nonsecure EPOCH counter

7.11.22 FLASH nonsecure EPOCH register (FLASH_EPOCHR_PRG)

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

HDPL0 level is necessary to modify this register contents.

This register is used to program values in corresponding option bits.

Address offset: 0x064

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPOCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **EPOCH[23:0]:** Non-volatile nonsecure EPOCH counter. The value programmed must be greater than the current value. Otherwise OPTCHANGEERR flag is raised.

7.11.23 FLASH secure EPOCH register (FLASH_SEPOCHR_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x0000 0001

Default value: 0x00FF FFFF (value in case of double ECC issue during OBL)

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x068

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPOCH[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **EPOCH[23:0]:** Non-volatile secure EPOCH counter

7.11.24 FLASH secure EPOCH register (FLASH_SEPOCHR_PRG)

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

HDPL0 level is necessary to modify this register contents.

This register is used to program values in corresponding option bits.

Address offset: 0x06C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPOCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **EPOCH[23:0]:** Non-volatile secure EPOCH counter. The value programmed must be greater than the current value. Otherwise OPTCHANGEERR flag is raised.

7.11.25 FLASH option status register 2 (FLASH_OPTSR2_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0xC300 00FC

Default value: 0xB400 00E0 (value in case of double ECC issue during OBL)

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x070

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZEN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM3_ECC	SRAM2_ECC	SRAM1_ECC	SRAM3_RST	SRAM2_RST	SRAM1_RST	Res.	Res.
								r	r	r	r	r	r		

Bits 31:24 **TZEN[7:0]**: TrustZone enable configuration bits
This bit enables the device is in TrustZone mode during an option byte change.
0xC3: TrustZone disabled
0xB4: TrustZone enabled.

Bits 23:8 Reserved, must be kept at reset value.

Bit 7 **SRAM3_ECC**: SRAM3 ECC detection and correction disable
0: SRAM3 ECC check enabled
1: SRAM3 ECC check disabled
Note that SRAM3_ECC is not available on the STM32G53x/54x devices.

Bit 6 **SRAM2_ECC**: SRAM2 ECC detection and correction disable
0: SRAM2 ECC check enabled
1: SRAM2 ECC check disabled

Bit 5 **SRAM1_ECC**: SRAM1 ECC detection and correction disable
0: SRAM1 ECC check enabled
1: SRAM1 ECC check disabled

Bit 4 **SRAM3_RST**: SRAM3 erase upon system reset
0: SRAM3 erased when a system reset occurs
1: SRAM3 not erased when a system reset occurs
Note that SRAM3_RST is not available on the STM32G53x/54x devices.

Bit 3 **SRAM2_RST**: SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs.

Bit 2 **SRAM1_RST**: SRAM1 erase upon system reset
0: SRAM1 erased when a system reset occurs
1: SRAM1 not erased when a system reset occurs

Bits 1:0 Reserved, must be kept at reset value.

7.11.26 FLASH option status register 2 (FLASH_OPTSR2_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

Address offset: 0x074

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZEN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM3_ECC	SRAM2_ECC	SRAM1_ECC	SRAM3_RST	SRAM2_RST	SRAM1_RST	Res.	Res.
								rw	rw	rw	rw	rw	rw		

Bits 31:24 **TZEN[7:0]**: TrustZone enable configuration bits
This bit enables the device is in TrustZone mode during an option byte change.
0xC3: TrustZone disabled
0xB4: TrustZone enabled

Bits 23:8 Reserved, must be kept at reset value.

Bit 7 **SRAM3_ECC**: SRAM3 ECC detection and correction disable
0: SRAM3 ECC check enabled
1: SRAM3 ECC check disabled
Note that SRAM3_ECC is not available on STM32G53x/54x devices.

Bit 6 **SRAM2_ECC**: SRAM2 ECC detection and correction disable
0: SRAM2 ECC check enabled
1: SRAM2 ECC check disabled

Bit 5 **SRAM1_ECC**: SRAM1 ECC detection and correction disable
0: SRAM1 ECC check enabled
1: SRAM1 ECC check disabled

Bit 4 **SRAM3_RST**: SRAM3 erase upon system reset
0: SRAM3 erased when a system reset occurs
1: SRAM3 not erased when a system reset occurs
Note that SRAM3_RST is not available on STM32G53x/54x devices.

Bit 3 **SRAM2_RST**: SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs.

Bit 2 **SRAM1_RST**: SRAM1 erase upon system reset
0: SRAM1 erased when a system reset occurs
1: SRAM1 not erased when a system reset occurs

Bits 1:0 Reserved, must be kept at reset value.

7.11.27 FLASH option status register 3 (FLASH_OPTSR3_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x00C3 C300

Default value: 0x00B4 C300 (value in case of double ECC issue during OBL)

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x078

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_LOCKOUT[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP_BANK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SWAP_LOCKOUT**: Configuration option specifies if the mapping of bank address space can be altered
0xC3: Swapping permitted
0xB4: Swapping disabled by configuration.
May only be modified in Open and Provisioning state

Bits 15:8 **SWAP_BANK**: Configuration option specifies the mapping of bank address space after reset
0xC3: Default mapping, Bank1 starts first
0xB4: Altered mapping, Bank2 starts first

Bits 7:0 Reserved, must be kept at reset value.

7.11.28 FLASH option status register 3 (FLASH_OPTSR3_PRG)

This register is secure when TrustZone® is active (RAZ, WI in nonsecure access) and nonsecure when TrustZone® is disabled. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

Address offset: 0x07C

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_LOCKOUT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP_BANK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SWAP_LOCKOUT**: Configuration option specifies if the mapping of bank address space can be altered
0xC3: Swapping permitted
0xB4: Swapping disabled by configuration.
May only be modified in Open and Provisioning state

Bits 15:8 **SWAP_BANK**: Configuration option specifies the mapping of bank address space after reset
0xC3: Default mapping, Bank1 starts first
0xB4: Altered mapping, Bank2 starts first

Bits 7:0 Reserved, must be kept at reset value.

7.11.29 FLASH nonsecure boot register (FLASH_BOOTR_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x0800 00C3

Default value: 0xFFFF FFB4 (value in case of double ECC issue during OBL)

This register reflects the current values of the corresponding option bits.

Address offset: 0x080

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BOOTADD[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOTADD[7:0]								BOOT_LOCK[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **BOOTADD[23:0]**: Non secure unique boot entry address
These bits reflect the Non secure UBE address

Bits 7:0 **BOOT_LOCK[7:0]:** Field locking the value of BOOTADD.
0xC3: BOOTADD can still be modified following their individual rules.
0xB4: BOOTADD is frozen.

7.11.30 FLASH nonsecure boot register (FLASH_BOOTR_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits.

Address offset: 0x084

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BOOTADD[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOTADD[7:0]								BOOT_LOCK[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 **BOOTADD[23:0]**: Non secure unique boot entry address
These bits allow configuring the Non secure BOOT address

Bits 7:0 **BOOT_LOCK[7:0]:** Field locking the BOOTADD setting.
0xC3: BOOTADD can still be modified following their individual rules.
0xB4: BOOTADD is frozen.

7.11.31 FLASH secure boot register (FLASH_SBOOTR_CUR)

This register is secure. It can be read only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x0C00 00C3

Default value: 0xFFFF FFB4 (value in case of double ECC issue during OBL)

This register reflects the current values of the corresponding option bits.

Address offset: 0x088

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SBOOTADD[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBOOTADD[7:0]								SBOOT_LOCK[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **SBOOTADD[23:0]**: Unique boot entry secure address
These bits reflect the Secure UBE address

Bits 7:0 **SBOOT_LOCK[7:0]:** Field locking the values of UBE and SBOOTADD settings.
0xC3: BOOT_UBE and SBOOTADD can still be modified following their individual rules.
0xB4: BOOT_UBE and SBOOTADD are frozen.

7.11.32 FLASH secure boot register (FLASH_SBOOTR_PRG)

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits.

Address offset: 0x08C

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SBOOTADD[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SBOOTADD[7:0]								SBOOT_LOCK[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 **SBOOTADD[23:0]**: Secure unique boot entry address.
These bits allow configuring the secure UBE address.

Bits 7:0 **SBOOT_LOCK[7:0]:** A field locking the values of UBE and SBOOTADD setting.
0xC3: BOOT_UBE and SBOOTADD can still be modified following their individual rules.
0xB4: BOOT_UBE and SBOOTADD are frozen.

7.11.33 FLASH nonsecure OTP block lock (FLASH_OTPBLR_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x0000 0000

Default value: 0xFFFF FFFF (value in case of double ECC issue during OBL)

This register reflects the current values of the corresponding option bits.

Address offset: 0x090

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCKBL[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCKBL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **LOCKBL[31:0]**: OTP block lock

Block *i* corresponds to OTP 16-bit word 32*i to 32**i* + 31.
LOCKBLi = 1 indicates that all OTP 16-bit words in OTP Block *i* are locked and attempt to program them results in WRPERR.
LOCKBLi = 0 indicates that all OTP 16-bit words in OTP Block *i* are not locked.
When one block is locked, it's not possible to remove the write protection.
Even if not locked, it is not possible to erase OTP words, only program virgin ones.

7.11.34 FLASH nonsecure OTP block lock (FLASH_OTPBLR_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits.

Address offset: 0x094

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCKBL[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCKBL[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **LOCKBL[31:0]**: OTP block lock

Block *i* corresponds to OTP 16-bit word 32*i to 32**i* + 31.
LOCKBLi = 1 indicates that all OTP 16-bit words in OTP Block *i* are locked and attempt to program them results in WRPERR.
LOCKBLi = 0 indicates that all OTP 16-bit words in OTP Block *i* are not locked.
When one block is locked, it is not possible to remove the write protection.
LOCKBL bits can be set if the corresponding bit in FLASH_OTPBLR_CUR is cleared.

7.11.35 FLASH bootloader interface selection (FLASH_BL_COM_CFG_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0xFFFF FFFF

Default value: 0xFFFF FFFF (value in case of double ECC issue during OBL)

This register reflects the current values of the corresponding option bits.

Address offset: 0x098

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BL_COM_CFG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BL_COM_CFG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BL_COM_CFG[31:0]**: Bootloader interface selection/configuration

This register is used to configure the set of interfaces on which the ST bootloader is active.
BL_COM_CFG[0] = 0 USART2 interface disabled for ST embedded bootloader.
BL_COM_CFG[0] = 1 USART2 interface enabled for ST embedded bootloader.
BL_COM_CFG[1] = 0 USART4 interface disabled for ST embedded bootloader.
BL_COM_CFG[1] = 1 USART4 interface enabled for ST embedded bootloader.
BL_COM_CFG[2] = 0 USART5 interface disabled for ST embedded bootloader.
BL_COM_CFG[2] = 1 USART5 interface enabled for ST embedded bootloader.
BL_COM_CFG[4] = 1 I2C2 interface disabled for ST embedded bootloader.
BL_COM_CFG[4] = 0 I2C2 interface enabled for ST embedded bootloader.
BL_COM_CFG[5] = 0 I2C3 interface disabled for ST embedded bootloader.
BL_COM_CFG[5] = 1 I2C3 interface enabled for ST embedded bootloader.
BL_COM_CFG[8] = 0 SPI1 interface disabled for ST embedded bootloader.
BL_COM_CFG[8] = 1 SPI1 interface enabled for ST embedded bootloader.
BL_COM_CFG[9] = 0 SPI2 interface disabled for ST embedded bootloader.
BL_COM_CFG[9] = 1 SPI2 interface enabled for ST embedded bootloader.
BL_COM_CFG[16] = 0 FDCAN1 interface disabled for ST embedded bootloader.
BL_COM_CFG[16] = 1 FDCAN1 interface enabled for ST embedded bootloader.
BL_COM_CFG[20] = 0 USB interface disabled for ST embedded bootloader.
BL_COM_CFG[20] = 1 USB interface enabled for ST embedded bootloader.
If the interface is enabled then it is used by the ST bootlader, corresponding GPIOs are used by ST bootloader.
If the interface is disabled then it is not used by the ST bootloader, corresponding GPIOs remain in default state.
For more information of bootloader interfaces and assigned GPIOs see the bootloader documentation (AN2606).

7.11.36 FLASH bootloader interface selection (FLASH_BL_COM_CFG_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits.

Address offset: 0x09C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BL_COM_CFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BL_COM_CFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **BL_COM_CFG[31:0]**: Bootloader interface selection/configuration

This register is used to configure the set of interfaces on which the ST bootloader is active.
BL_COM_CFG[0] = 0 USART2 interface disabled for ST embedded bootloader.
BL_COM_CFG[0] = 1 USART2 interface enabled for ST embedded bootloader.
BL_COM_CFG[1] = 0 USART4 interface disabled for ST embedded bootloader.
BL_COM_CFG[1] = 1 USART4 interface enabled for ST embedded bootloader.
BL_COM_CFG[2] = 0 USART5 interface disabled for ST embedded bootloader.
BL_COM_CFG[2] = 1 USART5 interface enabled for ST embedded bootloader.
BL_COM_CFG[4] = 1 I2C2 interface disabled for ST embedded bootloader.
BL_COM_CFG[4] = 0 I2C2 interface enabled for ST embedded bootloader.
BL_COM_CFG[5] = 0 I2C3 interface disabled for ST embedded bootloader.
BL_COM_CFG[5] = 1 I2C3 interface enabled for ST embedded bootloader.
BL_COM_CFG[8] = 0 SPI1 interface disabled for ST embedded bootloader.
BL_COM_CFG[8] = 1 SPI1 interface enabled for ST embedded bootloader.
BL_COM_CFG[9] = 0 SPI2 interface disabled for ST embedded bootloader.
BL_COM_CFG[9] = 1 SPI2 interface enabled for ST embedded bootloader.
BL_COM_CFG[16] = 0 FDCAN1 interface disabled for ST embedded bootloader.
BL_COM_CFG[16] = 1 FDCAN1 interface enabled for ST embedded bootloader.
BL_COM_CFG[20] = 0 USB interface disabled for ST embedded bootloader.
BL_COM_CFG[20] = 1 USB interface enabled for ST embedded bootloader.
If the interface is enabled then it is used by the ST bootlader, corresponding GPIOs are used by ST bootloader.
If the interface is disabled then it is not used by the ST bootloader, corresponding GPIOs remain in default state.
For more information of bootloader interfaces and assigned GPIOs see the bootloader documentation (AN2606).

7.11.37 FLASH secure block based registers for Bank1 (FLASH_SECBB1R)

These registers are secure. They can be read and written only by secure access. A nonsecure read/write access is RAZ/WI.

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC1[31:0]**: Secure/nonsecure 8 Kbytes flash Bank1 page attributes (y= 0 to 31)
0: page (y) in Bank1 is non secure.
1: page (y) in Bank1 is secure.
Note: For products with lower page count consider the extra upper bits reserved.

7.11.38 FLASH privilege block based registers for Bank1 (FLASH_PRIVBB1R)

These registers is privileged. They can be read and written only by a privileged access. These registers can be protected against nonsecure write access.

Address offset: 0x0C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV1[31:0]**: Privileged/unprivileged 8-Kbyte flash Bank1 page attribute (y = 0 to 31)
0: page (y) in Bank1 is unprivileged.
1: page (y) in Bank1 is privileged.
Note: For products with lower page count consider the extra upper bits reserved.

7.11.39 FLASH security watermark for Bank1 (FLASH_SECWM1R_CUR)

This read-only register is nonsecure, it can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x001F 0000

Default value: 0x001F 0000

It reflects the current values of the corresponding option bits.

Address offset: 0x0E0

Reset value: 0x00XX 00XX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
											r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **END[4:0]**: Bank1 security WM area 1 end page
Note: For products with lower page count consider the bits 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **STRT[4:0]**: Bank1 security WM area 1 start page
Note: For products with lower page count consider the bits 4 or 3 reserved.

7.11.40 FLASH security watermark for Bank1 (FLASH_SECWM1R_PRG)

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register. This register is used to program values in the corresponding option bits.

Address offset: 0x0E4

Reset value: 0x00XX 00XX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **END[4:0]**: Bank1 security WM area 1 end page
Note: For products with lower page count consider the bits 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **STRT[4:0]**: Bank1 security WM area 1 start page
Note: For products with lower page count consider the bits 4 or 3 reserved.

7.11.41 FLASH write page group protection for Bank1 (FLASH_WRP1R_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses, and can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0xFFFF FFFF

Default value: 0x0000 0000

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x0E8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPG1[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPG1[7:0]**: Bank1 page group protection option status byte
Each FLASH_WRP1R_CUR bit reflects the write protection status of the corresponding group of four consecutive pages in Bank1 (0: the group is write-protected; 1: the group is not write-protected)
Bit 0: Group embedding pages 0 to 3
Bit 1: Group embedding pages 4 to 7
Bit N: Group embedding pages 4 x N to 4 x N + 3
Bit 7: Group embedding pages 31 to 28
Note: For products with lower page count consider the extra bits reserved.

7.11.42 FLASH write page group protection for Bank1 (FLASH_WRP1R_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits.

Address offset: 0x0EC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPG1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPG1[7:0]**: Bank1 page group protection option status byte

Setting WRPG1 bits to 0 write protects the corresponding group of four consecutive pages in Bank1 (0: the group is write-protected; 1: the group is not write-protected)

Bit 0: Group embedding pages 0 to 3

Bit 1: Group embedding pages 4 to 7

Bit N: Group embedding pages 4 x N to 4 x N + 3

Bit 7: Group embedding pages 31 to 28

Note: For products with lower page count consider the bits representing unsupported pages reserved.

7.11.43 FLASH data page configuration Bank1 (FLASH_EDATA1R_CUR)

This register is nonsecure, can be read by both secure and nonsecure accesses. The register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x0000 0000

Default value: 0x0000 0000

This read-only register reflects the current values of the corresponding options bits.

Address offset: 0x0F0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[2:0]		
r													r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EN:** Bank1 flash high-cycle data enable
0: No flash high-cycle data area
1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **STRT[2:0]:**
STRT contains the start pages of the flash high-cycle data area in Bank1 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.
000: The last page of the Bank1 is reserved for flash high-cycle data
001: The two last pages of the Bank1 are reserved for flash high-cycle data
010: The three last pages of the Bank1 are reserved for flash high-cycle data
...
111: The eight last pages of the Bank1 are reserved for flash high-cycle data
Note: For products with lower EDATA page count consider the bit 2 reserved.

7.11.44 FLASH data page configuration Bank1 (FLASH_EDATA1R_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding options bits.

Address offset: 0xF4

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[2:0]		
rw													rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EN:** Bank1 flash high-cycle data enable
0: No flash high-cycle data area
1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **STRT[2:0]:**
STRT contains the start pages of the flash high-cycle data area in Bank1 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.
000: The last page of the Bank1 is reserved for flash high-cycle data
001: The two last pages of the Bank1 are reserved for flash high-cycle data
010: The three last pages of the Bank1 are reserved for flash high-cycle data
...
111: The eight last pages of the Bank1 are reserved for flash high-cycle data
Note: For products with lower EDATA page count consider the bit 2 reserved.

7.11.45 FLASH HDP Bank1 configuration (FLASH_HDP1R_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses, and can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register. This register is initially loaded with the values of the corresponding option bits.

Production value: 0x0000 0001

Default value: 0x001F 0000

Address offset: 0x0F8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
											r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **END[4:0]:** HDPL barrier end set in number of 8-Kbyte pages
Note: For products with lower page count consider the bit 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **STRT[4:0]:** HDPL barrier start set in number of 8-Kbyte pages
Note: For products with lower page count consider the bit 4 or 3 reserved.

7.11.46 FLASH HDP Bank1 configuration (FLASH_HDP1R_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure access, and can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

The register is accessible only in HDPL1. In HDPL2 or HDPL3 it is WI, RAZ.

This register is used to program values in the corresponding option bits.

Address offset: 0x0FC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **END[4:0]:** HDPL barrier end set in number of 8-Kbyte pages
Note: For products with lower page count consider the bit 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **STRT[4:0]:** HDPL barrier start set in number of 8-Kbyte pages
Note: For products with lower page count consider the bit 4 or 3 reserved.

7.11.47 FLASH ECC correction register (FLASH_ECCCORR)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses, and it can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ECCC	Res.	Res.	Res.	Res.	ECCCIE	OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Res.	Res.	Res.	Res.
	rw					rw	r	r	r	r	r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ECCC:** ECC correction set by hardware when single ECC error has been detected and corrected.
Cleared by writing 1.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **ECCCIE**: ECC single correction error interrupt enable bit
When ECCCIE bit is set to 1, an interrupt is generated when an ECC single correction error occurs during a read operation.
0: no interrupt generated when an ECC single correction error occurs
1: nonsecure interrupt generated when an ECC single correction error occurs

Bit 24 **OTP_ECC**: OTP ECC error bit
This bit is set to 1 when one single ECC correction occurred during the last successful read operation from the read-only/ OTP area. The address of the ECC error is available in ADDR_ECC bitfield.

Bit 23 **SYSF_ECC**: ECC fail for corrected ECC error in system flash memory
It indicates if system flash memory is concerned by ECC error.

Bit 22 **BK_ECC**: ECC fail bank for corrected ECC error
It indicates which bank is concerned by ECC error

Bit 21 **EDATA_ECC**: ECC fail for corrected ECC error in flash high-cycle data area
It indicates if flash high-cycle data area is concerned by ECC error.

Bit 20 **OBK_ECC**: Single ECC error corrected in flash OB Keys storage area. It indicates the OBK storage concerned by ECC error.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address

When an ECC error occurs (for single correction) during a read operation, the ADDR_ECC contains the address that generated the error. It is reset when the flag error is reset.

The flash interface programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved.

The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

7.11.48 FLASH ECC detection register (FLASH_ECCDETR)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	Res.	Res.	Res.	Res.	Res.	Res.	OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Res.	Res.	Res.	Res.
rw							r	r	r	r	r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ECCD:** ECC detection

Set by hardware when two ECC error has been detected.

When this bit is set, a NMI is generated.

Cleared by writing 1. Needs to be cleared in order to detect subsequent double ECC errors.

Bits 30:25 Reserved, must be kept at reset value.

Bit 24 **OTP_ECC**: OTP ECC error bit

This bit is set to 1 when double ECC detection occurred during the last read operation from the read-only/ OTP area. The address of the ECC error is available in ADDR_ECC bitfield.

Bit 23 **SYSF_ECC**: ECC fail for double ECC error in system flash memory

It indicates if system flash memory is concerned by ECC error.

Bit 22 **BK_ECC**: ECC fail bank for double ECC error

It indicates which bank is concerned by ECC error

Bit 21 **EDATA_ECC**: ECC fail double ECC error in flash high-cycle data area

It indicates if flash high-cycle data area is concerned by ECC error.

Bit 20 **OBK_ECC**: ECC fail double ECC error in flash OB Keys storage area. It indicates the OBK storage concerned by ECC error.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address

When an ECC error occurs (double detection) during a read operation, the ADDR_ECC contains the address that generated the error. It is reset when the flag error is reset.
The flash interface programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an double ECC error is saved.
The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

7.11.49 FLASH ECC data (FLASH_ECCDR)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses, and it can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA_ECC[15:0]**: ECC error data

When an double detection ECC error occurs on special areas with 6-bit ECC on 16-bit data (data area, read-only/OTP area), the failing data is read to this register.
By checking if it is possible to determine whether the failure was on a real data, or due to access to uninitialized memory.

7.11.50 FLASH secure block-based registers for Bank2 (FLASH_SECBB2R)

These registers are secure. They can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. The registers can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x1A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC2[31:0]**: Secure/nonsecure flash Bank2 page attribute (y = 0 to 31)
0: page (y) in Bank2 is non secure
1: page (y) in Bank2 is secure
Note: For products with lower page count consider the extra upper bits reserved.

7.11.51 FLASH privilege block-based registers for Bank2 (FLASH_PRIVBB2R)

These registers are privileged. They can be read and written only by a privileged access. The registers can be protected against nonsecure write access by watermark.

Address offset: 0x1C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV2[31:0]:** Privileged / non-privileged 8-Kbyte flash Bank2 page attribute (y = 0 to 31)
0: page (y) in Bank2 is unprivileged
1: page (y) in Bank2 is privileged
Note: For products with lower page count consider the extra upper bits reserved

7.11.52 FLASH security watermark for Bank2 (FLASH_SECWM2R_CUR)

This register is nonsecure. This register can be protected against unprivileged access when NSPRIpageV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x001F 0000

Default value: 0x001F 0000

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x1E0

Reset value: 0x00XX 00XX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
											r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **END[4:0]**: Bank2 security WM end page
Note: For products with lower page count consider the bit 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **STRT[4:0]**: Bank2 security WM area start page
Note: For products with lower page count consider the bit 4 or 3 reserved.

7.11.53 FLASH security watermark for Bank2 (FLASH_SECWM2R_PRG)

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits.

Address offset: 0x1E4

Reset value: 0x00XX 00XX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **END[4:0]**: Bank2 security WM end page
Note: For products with lower page count consider the bit 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **STRT[4:0]**: Bank2 security WM area start page
Note: For products with lower page count consider the bit 4 or 3 reserved.

7.11.54 FLASH write page group protection for Bank2 (FLASH_WRP2R_CUR)

This register is nonsecure, can be read by both secure and nonsecure accesses, and protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0xFFFF FFFF

Default value: 0x0000 0000

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x1E8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPG2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPG2[7:0]**: Bank2 page group protection option status byte

Each bit reflects the write protection status of the corresponding group of four consecutive pages in Bank2 (0: group is write-protected; 1: group is not write-protected)

Bit 0: Group embedding pages 0 to 3

Bit 1: Group embedding pages 4 to 7

Bit N: Group embedding pages 4 x N to 4 x N + 3

Bit 7: Group embedding pages 28 to 31

Note: For products with lower page count consider the extra upper bits reserved.

7.11.55 FLASH write page group protection for Bank2 (FLASH_WRP2R_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in the corresponding option bits.

Address offset: 0x1EC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPG2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WRPG2[7:0]**: Bank2 page group protection option status byte

Setting WRPGn2 bits to 0 write protects the corresponding group of four consecutive pages in Bank2 (0: group is write-protected; 1: group is not write-protected)

Bit 0: Group embedding pages 0 to 3

Bit 1: Group embedding pages 4 to 7

Bit N: Group embedding pages 4 x N to 4 x N + 3

Bit 7: Group embedding pages 28 to 31

Note: For products with lower page count consider the extra upper bits reserved.

7.11.56 FLASH data page configuration Bank2 (FLASH_EDATA2R_CUR)

This register is nonsecure. It can be read by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Production value: 0x0000 0000

Default value: 0x0000 0000

This read-only register reflects the current values of the corresponding options bits.

Address offset: 0x1F0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[2:0]		
r													r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EN:** Bank2 flash high-cycle data enable
0: No flash high-cycle data area
1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **STRT[2:0]:**
STRT contains the start pages of the flash high-cycle data area in Bank2 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.
000: The last page of the Bank2is reserved for flash high-cycle data
001: The two last pages of the Bank2 are reserved for flash high-cycle data
010: The three last pages of the Bank2 are reserved for flash high-cycle data
...
111: The eight last pages of the Bank2 are reserved for flash high-cycle data

Note: For products with lower page count consider the bit 2 reserved.

7.11.57 FLASH data page configuration Bank2 (FLASH_EDATA2R_PRG)

This register is nonsecure. It can be read and written by both secure and nonsecure accesses, and protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register. It is used to program values in the corresponding options bits.

Address offset: 0x1F4

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[2:0]		
rw													rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EN:** Bank2 flash high-cycle data enable
0: No flash high-cycle data area
1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **STRT[2:0]:**
STRT contains the start pages of the flash high-cycle data area in Bank2 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.
000: The last page of the Bank2 is reserved for flash high-cycle data.
001: The two last pages of the Bank2 are reserved for flash high-cycle data.
010: The three last pages of the Bank2 are reserved for flash high-cycle data.
...
111: The eight last pages of the Bank2 are reserved for flash high-cycle data.
Note: For products with lower page count consider the bit 2 reserved.

7.11.58 FLASH HDP Bank2 configuration (FLASH_HDP2R_CUR)

This register is initially loaded with the values of the corresponding option bits.

Production value: 0x0000 0001

Default value: 0x001F 0000

This register is nonsecure. It can be read by both secure and nonsecure accesses, and protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x1F8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
											r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **END[4:0]:** HDPL barrier end set in number of 8-Kbyte pages
Note: For products with lower page count consider the bit 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **STRT[4:0]:** HDPL barrier start set in number of 8-Kbyte pages
Note: For products with lower page count consider the bit 4 or 3 reserved.

7.11.59 FLASH HDP Bank2 configuration (FLASH_HDP2R_PRG)

This register is nonsecure. It can be read and written only by both secure and nonsecure accesses. This register can be protected against unprivileged access when PRIV = 1 in the FLASH_PRIVCFGR register.

Register is accessible only in HDPL1. In HDPL2 or HDPL3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

Address offset: 0x1FC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **END[4:0]:** HDPL barrier end set in number of 8-Kbyte pages
Note: For products with lower page count consider the bit 20 or 19 reserved.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **STRT[4:0]:** HDPL barrier start set in number of 8-Kbyte pages
Note: For products with lower page count consider the bit 4 or 3 reserved.

7.12 FLASH register map and reset values

Table 80. Register map and reset value table

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTEN	Res.	Res.	WRHIGHFREQ [1:0]		LATENCY [3:0]			
	Reset value																								0			0	1	0	0	1	0
0x004	FLASH_KEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x008	FLASH_ SKEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	FLASH_OPTKEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	FLASH_ OBKKEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	FLASH_ SOBKKEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	FLASH_OPSR	CODE_OP [2:0]			Res.	Res.	Res.	Res.	OTP_OP	SYSF_OP	BK_OP	DATA_OP	Res.	ADDR_OP[19:0]																			
	Reset value	X	X	X					X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x01C	FLASH_OPTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	OPTLOCK
	Reset value																															0	1
0x020	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGERR	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDBNE	PUF_STATE	WBNE	BSY
	Reset value									0	0	0	0	0	0	0	0														0	0	0	0
0x024	FLASH_SSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBNE	PUF_STATE	WBNE	BSY
	Reset value										0	0	0	0	0	0	0														X	X	X	X
0x028	FLASH_CR	BKSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERRIE	OBKWIE	OBKIE	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE	MER	Res.	PUF_LAUNCH	Res.	Res.	PNB[4:0]						STRT	FW	BER	PER	PG	LOCK
	Reset value	0								0	0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	1

Table 80. Register map and reset value table (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	FLASH_SCR	BKSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBKWIE	OBKERRIE	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE	MER	Res.	PUF_LAUNCH	Res.	Res.	PNB[4:0]					STRT	FW	BER	PER	PG	LOCK
	Reset value	0									0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	1
0x030	FLASH_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERR	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0																
0x034	FLASH_ SCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0	0	0	0	0																
0x03C	FLASH_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SPRIV
	Reset value																															0	0
0x040	FLASH_ OBKCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_OFFSET[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALT_SECT_ERASE	ALT_SECT	SWAP_SECT_REQ	LOCK
	Reset value								1	1	1	1	1	1	1	1	1													0	0	0	0
0x044	FLASH_ SOBKCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_OFFSET[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALT_SECT_ERASE	ALT_SECT	SWAP_SECT_REQ	LOCK
	Reset value								1	1	1	1	1	1	1	1	1													0	0	0	0
0x048	FLASH_HDPEXTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			HDP2_EXT[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			HDP1_EXT[4:0]				
	Reset value												0	0	0	0	0												0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	FLASH_ OPTSR_CUR		Res.	BOOT_UBE[7:0]								IWDG_STDBY	IWDG_STOP	Res.	Res.	Res.	Res.	PRODUCT_STATE [7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Res.	Res.	Res.
	Reset value			X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X		X	X			
0x054	FLASH_ OPTSR_PRG		Res.	BOOT_UBE[7:0]								IWDG_STDBY	IWDG_STOP	Res.	Res.	Res.	Res.	PRODUCT_STATE [7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	Res.	Res.	Res.
	Reset value			X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X		X	X			

Table 80. Register map and reset value table (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	Reserved	Reserved																															
0x05C	Reserved	Reserved																															
0x060	FLASH_ EPOCHR_CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH[23:0]																							
	Reset value									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x064	FLASH_ EPOCHR_PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH[23:0]																							
	Reset value									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x068	FLASH_ SEPOCHR_CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH[23:0]																							
	Reset value									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x06C	FLASH_ SEPOCHR_PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH[23:0]																							
	Reset value									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x070	FLASH_ OPTSR2_CUR	TZEN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM3_ECC	SRAM2_ECC	SRAM1_ECC	SRAM3_RST	SRAM2_RST	SRAM1_RST	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X																	X	X	X	X	X	X		
0x074	FLASH_ OPTSR2_PRG	TZEN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM3_ECC	SRAM2_ECC	SRAM1_ECC	SRAM3_RST	SRAM2_RST	SRAM1_RST	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X																	X	X	X	X	X	X		
0x080	FLASH_ BOOTR_CUR	BOOTADD[23:0]																								BOOT_LOCK[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x084	FLASH_ BOOTR_PRG	BOOTADD[23:0]																								BOOT_LOCK[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x088	FLASH_ SBOOTR_CUR	SBOOTADD[23:0]																								SBOOT_LOCK[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08C	FLASH_ SBOOTR_PRG	SBOOTADD[23:0]																								SBOOT_LOCK[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x090	FLASH_ OTPBLR_CUR	LOCKBL[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x094	FLASH_ OTPBLR_PRG	LOCKBL[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x098	FLASH_BL_COM_ CFG_CUR	BL_COM_CFG[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x09C	FLASH_BL_COM_ CFG_PRG	BL_COM_CFG[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 80. Register map and reset value table (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A0	FLASH_ SECBB1R	SEC1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C0	FLASH_ PRIVBB1R	PRIV1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D0 - 0x0DC	Reserved	Reserved																															
0x0E0	FLASH_SECWM1R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x0E4	FLASH_SECWM1R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END1[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x0E8	FLASH_WRP1R_ CUR	WRPG1[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0EC	FLASH_WRP1R _PRG	WRPG1[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0F0	FLASH_EDATA1R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[2:0]		
	Reset value																	X													X	X	X
0x0F4	FLASH_EDATA1R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[2:0]		
	Reset value																	X													X	X	X
0x0F8	FLASH_HDP1R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x0FC	FLASH_HDP1R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x100	FLASH_ECCCORR	Res.	ECCC	Res.	Res.	Res.	Res.	ECCCIE	OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Res.	Res.	Res.	Res.	ADDR_ECC[15:0]															
	0x0000 0000		0					0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	FLASH_ECCDETR	ECCD	Res.	Res.	Res.	Res.	Res.	Res.	OTP_ECC_FAIL	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Res.	Res.	Res.	Res.	ADDR_ECC[15:0]															
	Reset value	0							0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	FLASH_ECCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA_ECC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 80. Register map and reset value table (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1A0	FLASH_ SECBB2R	SEC2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A0	FLASH_ PRIVBB2R	PRIV2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D0 - 0x1DC	Reserved	Reserved																															
0x1E0	FLASH_SECWM2R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x1E4	FLASH_SECWM2R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END1[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x1E8	FLASH_WRP2R _CUR	WRPG2[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x1EC	FLASH_WRP2R _PRG	WRPG2[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x1F0	FLASH_EDATA2R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[2:0]		
	Reset value																	X													X	X	X
0x1F4	FLASH_EDATA2R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[2:0]		
	Reset value																	X													X	X	X
0x1F8	FLASH_HDP2R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X
0x1FC	FLASH_HDP2R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			END[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[4:0]				
	Reset value												X	X	X	X	X												X	X	X	X	X

Refer to *Section 2.3* for the register boundary addresses.

8 Instruction cache (ICACHE)

8.1 ICACHE introduction

The instruction cache (ICACHE) is implemented on the C-AHB code bus of the Cortex®-M33 processor to improve performance when fetching instructions and data from internal and external memories.

Some specific features, like dual master ports, hit-under-miss, and critical-word-first refill policy, result in close to zero-wait-state performance in most use cases.

8.2 ICACHE main features

The main features of ICACHE are listed below:

- Bus interface
  - One 32-bit AHB slave port, the execution port (input from Cortex®-M33 C-AHB code interface)
  - Two AHB master ports: master1 and master2 ports, with outputs to Fast and Slow buses of main AHB bus matrix, respectively(master2 port is available only on STM32G55x/56x devices)
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Dual master(only on STM32G55x/56x devices) access: feature used to decouple the traffic according to targeted memory. For example, the ICACHE assigns fast traffic (addressing flash memories and SRAMs) to the AHB master1 port, and slow traffic (addressing external memories) to the AHB master2 port, thus preventing processor stalls on line refills from external memories. This allows ISR (interrupt service routine) fetching on internal flash memory to take place in parallel with a cache line refill from external memories.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, **w**, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile
- Memory address remap
  - Possibility to remap input address falling into up to four memory regions (used to remap aliased code in external memories to the code region, for execution from the C-AHB code interface)
- Replacement and refill
  - pLRU-t (pseudo-least-recently-used, based on binary tree) replacement policy, algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls

  - Possibility to configure burst type of AHB memory transaction for remapped regions: INCRw or WRAPw (size w aligned on cache line size)
- ECC hardware support: single-bit and double-bit error detection (DED)
- Performance counters

  The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)
- Error management
  - Possibility to detect an unexpected cacheable write access, to flag an error, and optionally, to raise an interrupt
- TrustZone® security support
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, noninterruptible

8.3 ICACHE implementation

Table 81. ICACHE features for STM32G55x/56x

Feature	ICACHE1
Number of ways	2
Cache size	8 Kbytes
Cache line width	16 bytes
Range granularity of memory regions to be remapped	2 Mbytes
Number of regions to remap	4
Data size of AHB slave interface	32 bits
Data size of AHB fast master1 interface	128 bits
Data size of AHB slow master2 interface	32 bits
ECC support	Yes

Table 82. ICACHE features for STM32G53x/54x

Feature	ICACHE1
Number of ways	2
Cache size	8 Kbytes
Cache line width	16 bytes
Range granularity of memory regions to be remapped	2 Mbytes
Number of regions to remap	4
Data size of AHB slave interface	32 bits
Data size of AHB fast master1 interface	128 bits
Data size of AHB slow master2 interface	-
ECC support	Yes

8.4 ICACHE functional description

The purpose of this block is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE manages only cacheable read transactions, and does not manage cacheable write transactions.

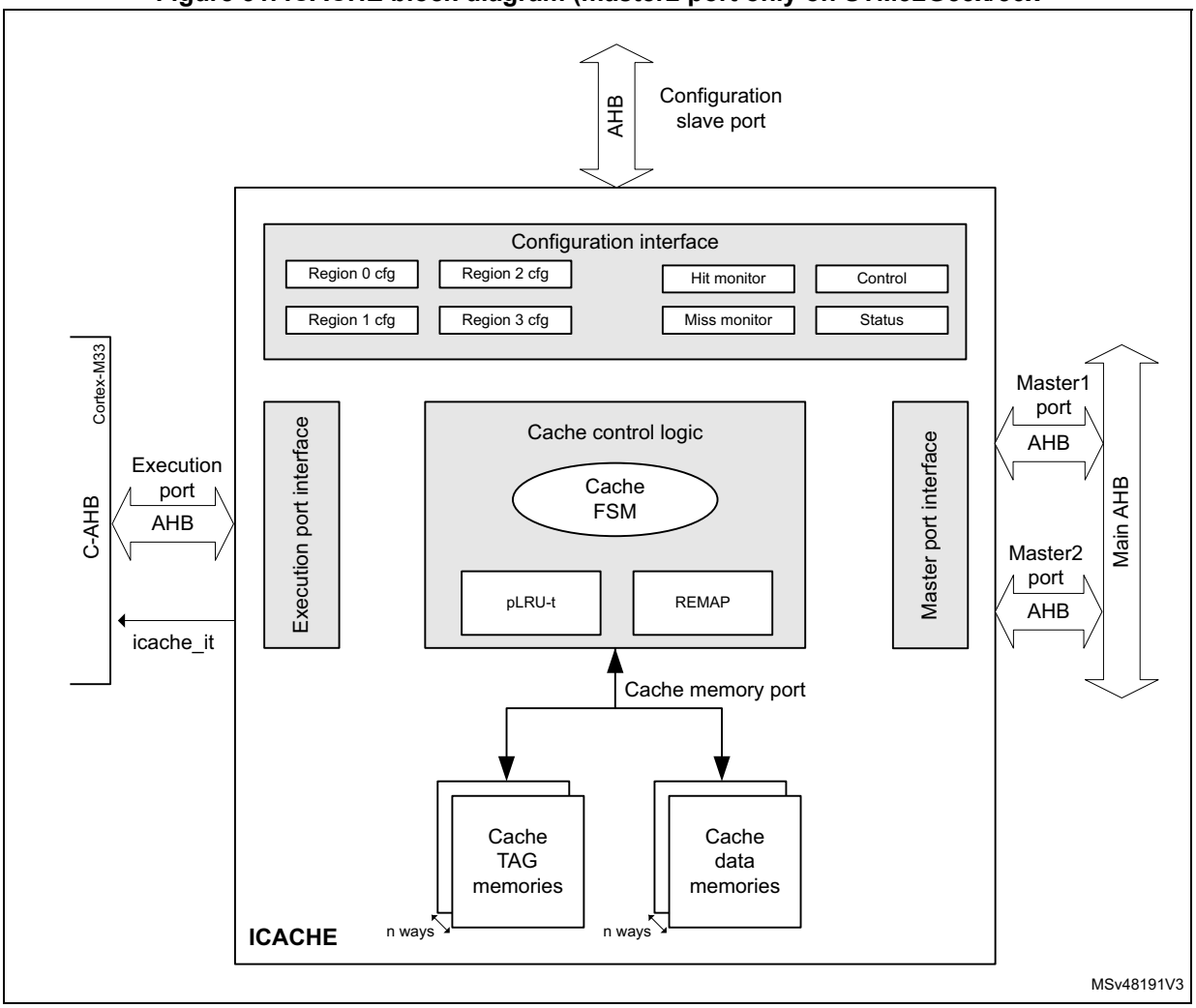
The noncacheable transactions (both read and write) bypass the ICACHE.

For the error management purpose, if a write cacheable transaction is presented (this happens only in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

The ICACHE also implements the hardware support for ECC, with a single-bit and double-bit error detection (DED; see *Section 8.4.12: ICACHE ECC support* for more details on ECC feature).

8.4.1 ICACHE block diagram

Figure 31. ICACHE block diagram (master2 port only on STM32G55x/56x



8.4.2 ICACHE reset and clocks

The ICACHE is clocked on the Cortex®-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 and also all TAG addresses, ECC bit-fields and data reset to 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (= 0).

Note: When disabled, the ICACHE is bypassed, except the remapping mechanism that is still functional: slave input requests (remapped or not) are forwarded to the master port(s).

8.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags indicating which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits and also TAG addresses, ECC bit-fields and data are reset in any of the below cases:

- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see *Table 83* for B and W definitions):

- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 32* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

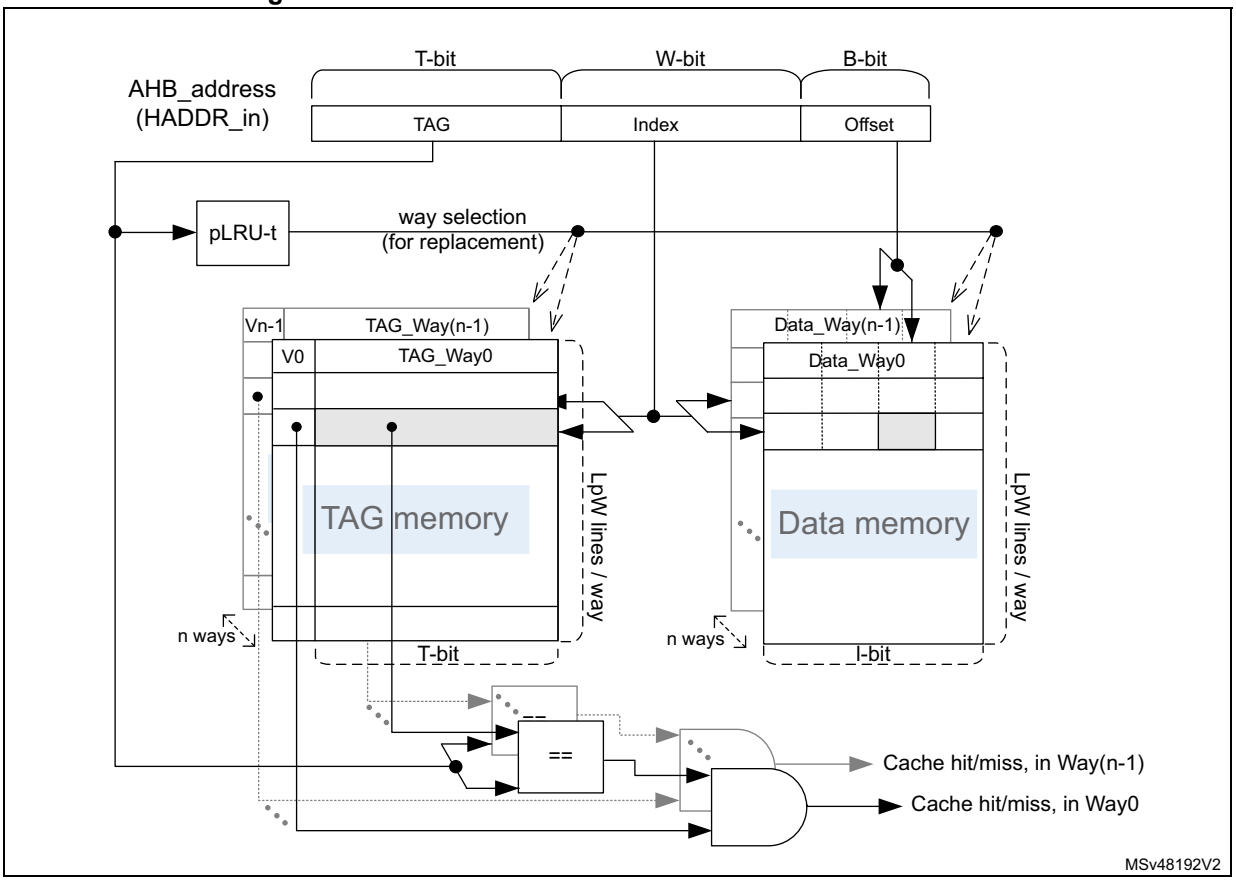
Table 83. TAG memory dimensioning parameters for n-way set associative operating mode (default)

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit

Table 83. TAG memory dimensioning parameters for n-way set associative operating mode (default) (continued)

Parameter	Value	Example
Number of cache lines	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 32. ICACHE TAG and data memories functional view



8.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

Table 84. TAG memory dimensioning parameters for direct-mapped cache mode

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	1	1
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / L lines	512 lines
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, remapping, invalidation) remain the same in the direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in *Section 8.4.8*): only one way (the unique one) is possible for any data refill.

8.4.5 ICACHE enable

To activate the ICACHE, the EN bit in ICACHE_CR must be set to 1.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave to the master ports in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate (and initialize) procedure before enabling the ICACHE. This procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether the ICACHE is enabled or not, if the input transaction address falls into the memory regions defined and enabled in ICACHE_CRRx (see *Figure 33*).

The ICACHE is, by default, disabled at boot.

8.4.6 Cacheable and noncacheable traffic

The ICACHE is developed for the Cortex®-M33 core. It is placed on the C-AHB bus, and thus caches the code memory region, ranging from address 0x0000 0000 to 0x1FFF FFFF of the memory map.

To make other memory regions cacheable, the ICACHE supports a memory region remapping feature, used to define up to four external memory regions, whose addresses have an alias in the code region. Addressing these memory regions through their code alias address allows the memory request to be routed to the C-AHB bus, and to be managed by the ICACHE.

Any external memory space physically mapped at an address in the range [0x6000 0000:0xAFFF FFFF] can be aliased with an address in the range [0x0000 0000:0x07FF FFFF] or [0x1000 0000:0x1FFF FFFF].

For a given memory request in the code region, the ICACHE implements the address remapping functionality first. If aliased, it is the remapped address, which is then cached, and, if needed, provided to the master port to address the main AHB bus matrix. The destination physical address does not need further manipulation on the AHB bus.

The remapping functionality is available also for noncacheable traffic, and when the cache is disabled.

Additional details on address remapping are provided in *Section 8.4.7*.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 85*. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 85. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (read or write), the ICACHE is bypassed. The AHB transaction is propagated unchanged to the master output port, except the transaction address, which may be modified due to the address remapping feature (see *Section 8.4.7*).

The bypass, and eventual remap logic, does not increase the latency of the access to the targeted memory.

In the case of a cacheable access, the ICACHE behaves as explained in *Section 8.4.8*.

Cacheable memory regions are defined and programmed by the user in the MPU responsible for the generation of the AHB attribute signals (HPROT[6:0]) for any transaction addressing a given region.

Table 86 summarizes programmable configurations of various memories.

Table 86. Memory configurations

Memory	Cacheable (MPU programming)	Remapped in the ICACHE (ICACHE_CRRx programming)
Flash	Yes or no	Not required
SRAM	Not recommended	Not required
External memories	Yes	Required
	No	Required if the user wants code in external memories fetched on C-AHB bus (else on S-AHB bus)

8.4.7 Address remapping

The ICACHE allows an alias address to be defined in the code region for up to four external memory regions.

The address remapping is applied on the code alias address, transforming it into the destination external physical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx (x = 0 to 3, number of remapped regions). This programming can be done only when the ICACHE is disabled.

Each region x can be individually enabled with REN in ICACHE_CRRx. Once enabled, the remap operation occurs even if the ICACHE is disabled, or if the transaction is noncacheable.

Remap regions can have different size: each can be programmed in RSIZE of its ICACHE_CRRx. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum of 2 Mbytes, and a maximum of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given code subregion base-address, and the modification of this address into its (remapped) physical address, as follows:

- HADDR_in belongs to region x if HADDR_in[31:RI] = 000:BASEADDR[28:RI], where:
  - 000:BASEADDR is the code subregion base-address programmed in BASEADDR of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = $\log_2$ (region size) with a minimum value of 21 (for a 2-Mbyte region) and a maximum value of 27 (for a 128-Mbyte region).
- If the region x is enabled, the master port output AHB address (HADDR_out) is composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs

Figure 33 describes the matching and the output address generation.

Figure 33. ICACHE remapping address mechanism

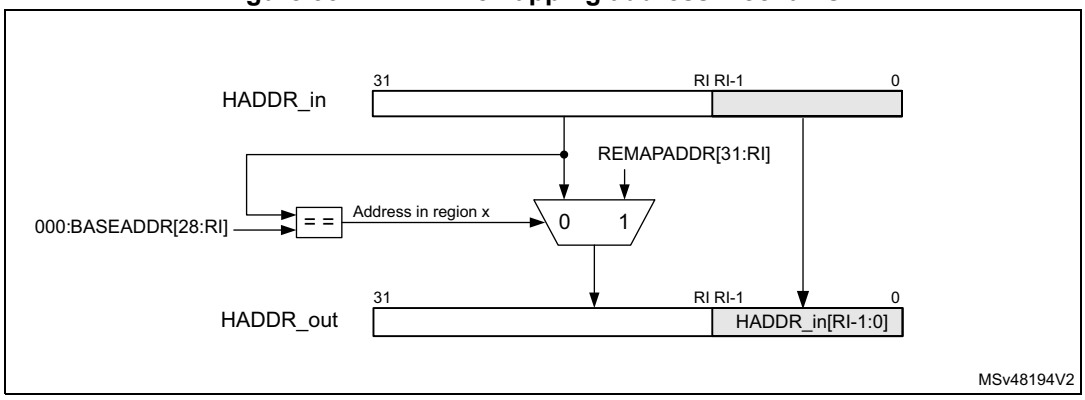


Table 87 summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending upon RSIZE.

Table 87. ICACHE remap region size, base address, and remap address

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
2	8	11
4	7	10
8	6	9
16	5	8
32	4	7
64	3	6
128	2	5

Care must be taken while programming BASEADDR and REMAPADDR in ICACHE_CRRx: if the programmed value is bigger than expected (number of MSBs, see *Table 87*), the unneeded extra LSBs are ignored.

Typical remapping example: a 128-Mbyte external RAM region physically located in the external address range [0x9000 0000:0x97FF FFFF], remapped in the code section range [0x1800 0000:0x1FFF FFFF]:

- REMAPADDR[31:21] = 0x480
- BASEADDR[28:21] = 0xC0
- HADDR_in[31:27] is compared to 000:BASEADDR[28:27], while HADDR_in and BASEADDR[26:21] are ignored.

If the comparison matches:

- HADDR_out[31:27] gets REMAPADDR[31:27] (in place of HADDR_in[31:27])
- HADDR_out[26:0] gets HADDR_in[26:0]

The software can program the kind of AHB burst generated by the ICACHE master ports on the bus matrix (for cache line refill), by setting HBURST in ICACHE_CRRx with:

- WRAP for remapped external memories accessed through interfaces that can support WRAP burst mode, providing the benefit of the critical-word-first feature performance:
  - WRAP burst size = cache line size
  - WRAP burst start address = word address of the first data requested by the core
- INCR: INCR burst mode for external memories accessed through the interfaces that do not support WRAP burst mode (losing the benefit of critical-word-first feature):
  - INCR burst size = cache line size
  - INCR burst start address = address aligned on the boundary of the cache line containing the requested word.

Note: Coherency is needed when programming the SAU (secure attribution unit) and the MPU (memory protection unit) attributes for both the external regions and their aliased code subregions.

8.4.8 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex®-M33, it checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy ensures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).

  The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words) if no address remap occurs. If an address remap occurs, the kind of burst depends upon the HBURST programmed in corresponding ICACHE_CRRx.

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port selected for the line refill.
- The address is not present in TAG memory, but belongs to the refill burst from the main memory currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache-line refill. The ICACHE can provide the requested data as soon as data are available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
- In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

8.4.9 Dual-master cache

The ICACHE implements a dual-port AHB master on the main AHB bus matrix: master1 and master2 ports(master2 port is available only on STM32G55x/56x devices). This is used to split the traffic going to different destination memories.

The nonremapped traffic goes systematically to master1 port. The remapped traffic can be routed on the master2 port by programming MSTSEL in ICACHE_CRRx (on a region basis).

The code can typically be fetched as follows:

- internal flash memory and internal SRAM on master1 port (Fast bus)
- external flash memory/RAM on master2 port (Slow bus)

For systems not implementing external memories, the traffic to the internal flash memory can be decoupled from the traffic to the internal SRAM (when remapped by the ICACHE). This feature is used to prevent further processor stalls on misses.

Alongside with hit-under-miss, this dual-master feature allows the processor to have an alternative path in case of fetching from different memories.

8.4.10 ICACHE security

The ICACHE implements an Armv8-M TrustZone.

ICACHE configuration registers are protected at system level.

8.4.11 ICACHE maintenance

The software can invalidate (and initialize) the whole content of the ICACHE by programming CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). Each TAG addresses, ECC bit-fields and data are also reset to 0 during cache invalidation to prevent unknown values and potential ECC errors detection at the next cache line read access. CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits and also TAG addresses, ECC bit-fields and data reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.

8.4.12 ICACHE ECC support

An error correction code (ECC) hardware is implemented in the ICACHE to detect errors when a cache line is read.

This ICACHE ECC hardware enables the detection of single-bit and double-bit errors, and is then called DED ECC.

The ECC feature is implemented by the addition (in TAG memory) of an extra ECC bitfield to each ICACHE cache line, in order to provide some redundancy in the TAG + data memories content.

The ECC bitfield size depends on the cache line size (TAG and data).

When the ECC function is enabled (by setting ECCEN = 1 in ICACHE_CR), the ECC bitfield:

- is encoded by the ECC encoder, based on both the TAG (including address and valid bits) and the data to write in a given cache line (when the ICACHE is refilled from the main memory, on cache miss occurrence)
- is written along with the cache line TAG
- is read along with the TAG of any cache line (when the cache line TAG is accessed to check if its address matches or not with the address of the incoming ICACHE fetch request)
- is decoded along with the read cache line (TAG and data), in order to detect if one or two bits are erroneous in this TAG and/or data.

If no ECC error is detected, the ICACHE processes the request as usual (hit or miss).

If the ECC decoder detects a single-bit or a double-bit error on a given cache line (read for the TAG comparison process), the ICACHE:

- considers the cache access as a miss (even if the TAG read matches the address of the incoming request, because this TAG may contain errors).
- invalidates (and resets) the erroneous cache line (this can be considered as an indirect correction of the cache content)
- processes this miss as a usual miss, by refilling a cache line from the main memory and then providing a safe data to the fetch requester.
- reports the single-bit or double-bit error detection to allow the system to measure some ICACHE memories corruption (soft error) and/or aging (see *Section 8.6: ICACHE error management and interrupts* for more details on the related error flags).

As the TAG (and data) are read, in parallel, in all the ICACHE ways, an ECC decoder is implemented for each way.

Table 88 summarizes the behavior of the ICACHE when the ECC feature is enabled.

Table 88. ICACHE read requests management (when ECC is enabled)

ECC error detected on cache line (TAG address/valid bits or data)	Decision and operation	ECC error reporting
No ECC error	Hit or Miss	No
Single-bit ECC error	Miss (erroneous cache line invalidated)	ECC single-bit error flag
Double-bit ECC error		ECC double-bit error flag

When the ECC function is disabled (by setting ECCEN = 0 in ICACHE_CR), the ECC bitfield is neither encoded in the cache lines (not written), nor decoded on the cache lines read, and the ICACHE behaves as if the ECC feature is not implemented (including the ECC single-bit error and double-bit error flags remain clear).

8.4.13 ICACHE performance monitoring

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.

  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 8.4.8*).
- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.

  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

8.4.14 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

The code remapping at boot is not needed for a Cortex®-M33 as it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

8.5 ICACHE low-power modes

At device level, using the ICACHE reduces the power consumption by fetching instructions from the internal ICACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with lower performance profile (in terms of hit ratio) and stringent power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 32*). The power consumption is reduced by accessing, for each request, only the necessary cut of TAG and data memories. The cache effect still improves fetch performance, even if for most code execution, it is less efficient than with an n-way set associative cache mode.

8.6 ICACHE error management and interrupts

If an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

When the ECC function is enabled (by setting ECCEN = 1 in ICACHE_CR), two ECC error flags can be set in ICACHE_SR:

- the ECC single-bit error flag, SERRF
- the ECC double-bit error flag, DERRF

and an interrupt is then generated if the corresponding interrupt enable bit is set (SERRIE = 1, respectively DERRIE = 1, in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

Table 89. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF in ICACHE_SR	ERRIE in ICACHE_IER	Set CERRF to 1 in ICACHE_FCR
	ECC single-bit error	SERRF in ICACHE_SR	SERRIE in ICACHE_IER	Set CSERRF to 1 in ICACHE_FCR
	ECC double-bit error	DERRF in ICACHE_SR	DERRIE in ICACHE_IER	Set CDERRF to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF in ICACHE_SR	BSYENDIE in ICACHE_IER	Set CBSYENDF to 1 in ICACHE_FCR

The ICACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from the master1 or master2 port back to the execution port.

8.7 ICACHE registers

8.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCEN	Res.	WAYSEL	CACHEINV	EN
											rw		rw	w	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **MISSMRST**: miss monitor reset

0: release the cache miss monitor reset (needed to enable the counting)

1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset

0: release the cache miss monitor reset (needed to enable the counting)

1: reset cache hit monitor

Bit 17 **MISSMEN**: miss monitor enable

0: cache miss monitor switched off. Stopping the monitor does not reset it.

1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable

0: cache hit monitor switched off. Stopping the monitor does not reset it.

1: cache hit monitor enabled

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **ECCEN**: ECC enable

ECCEN can be modified (by software) only when the cache is disabled (EN = 0).

0: ECC encoder and decoder disabled

1: ECC encoder and decoder enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection

This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).

0: direct mapped cache (1-way cache)

1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation

Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.

0: no effect

1: invalidate entire cache (all cache lines valid bit and TAG/ECC/data = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

8.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DERRF	SERRF	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
						rw	rw						r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **DERRF**: ECC double-bit error flag
DERRF can be set to 1 by software for debug purpose; writing 0 has no effect (clearing through CDERRF in ICACHE_FCR)
0: no ECC double-bit error detected
1: an ECC double-bit error was detected

Bit 8 **SERRF**: ECC single-bit error flag
SERRF can be set to 1 by software for debug purpose; writing 0 has no effect (clearing through CSERRF in ICACHE_FCR)
0: no ECC single-bit error detected
1: an ECC single-bit error was detected

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYENDF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

8.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DERRIE	SERRIE	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
						rw	rw						rw	rw	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **DERRIE**: interrupt enable on ECC double-bit error detection
Set by software to enable an interrupt generation in case of ECC double-bit error detection
0: interrupt disabled on ECC double-bit error detection
1: interrupt enabled on ECC double-bit error detection

Bit 8 **SERRIE**: interrupt enable on ECC single-bit error detection
Set by software to enable an interrupt generation in case of ECC single-bit error detection
0: interrupt disabled on ECC single-bit error detection
1: interrupt enabled on ECC single-bit error detection

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **ERRIE**: interrupt enable on cache error
Set by software to enable an interrupt generation in case of cache functional error (cacheable write access)
0: interrupt disabled on error
1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end
Set by software to enable an interrupt generation at the end of a cache invalidate operation.
0: interrupt disabled on busy end
1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

8.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CDERRF	CSERRF	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
						w	w						w	w	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CDERRF**: clear ECC double-bit error flag
Set by software.
0: no effect
1: clears DERRF flag in ICACHE_SR

Bit 8 **CSERRF**: clear ECC single-bit error flag
Set by software.
0: no effect
1: clears SERRF flag in ICACHE_SR

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **CERRF**: clear cache error flag
Set by software.
0: no effect
1: clears ERRF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag
Set by software.
0: no effect
1: clears BSYENDF flag in ICACHE_SR.

Bit 0 Reserved, must be kept at reset value.

8.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

8.7.6 ICACHE miss monitor register (ICACHE_MMONR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISSMON[15:0]**: cache miss monitor counter

8.7.7 ICACHE region x configuration register (ICACHE_CRRx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define an alias address in the code region for other regions, making them cacheable.

BASEDADDR and REMAPADDR fields are write locked (read only) when EN = 1 in ICACHE_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]										
rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	BASEADDR[28:21]							
rw				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HBURST**: output burst type for region x
0: WRAP
1: INCR

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **MSTSEL**: AHB cache master selection for region x
0: no action (master1 selected by default)
1: master2 selected
Note: MSTSEL is not available on the STM32G53x/54x devices.

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **REMAPADDR[31:21]**: remapped address for region x
This field replaces the alias address defined by BASEADDR field.
The only useful bits are [31:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

Bit 15 **REN**: enable for region x
0: disabled
1: enabled

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:9 **RSIZE[2:0]**: size for region x
000: reserved
001: 2 Mbytes
010: 4 Mbytes
011: 8 Mbytes
100: 16 Mbytes
101: 32 Mbytes
110: 64 Mbytes
111: 128 Mbytes

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **BASEADDR[28:21]**: base address for region x
This alias address is replaced by REMAPADDR field.
The only useful bits are [28:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

8.7.8 ICACHE register map

Table 90. ICACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ICACHE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCEN	Res.	WAYSEL	CACHEINV	EN
	Reset value													0	0	0	0												0		1	0	0
0x004	ICACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERRF	SERRF	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
	Reset value																							0	0						0	0	1
0x008	ICACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERRIE	SERRIE	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
	Reset value																							0	0						0	0	
0x00C	ICACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDERRF	CSERRF	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
	Reset value																							0	0						0	0	
0x010	ICACHE_HMONR	HITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	ICACHE_MMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved	Reserved																															
0x020	ICACHE_CRR0	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]		Res.	BASEADDR[28:21]								
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x024	ICACHE_CRR1	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]		Res.	BASEADDR[28:21]								
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x028	ICACHE_CRR2	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]		Res.	BASEADDR[28:21]								
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0
0x02C	ICACHE_CRR3	HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]											REN	Res.	Res.	Res.	RSIZE [2:0]		Res.	BASEADDR[28:21]								
	Reset value	0			0		0	0	0	0	0	0	0	0	0	0	0	0				0	0	1		0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

9 Data cache (DCACHE)

9.1 DCACHE introduction

The data cache (DCACHE) is introduced, only for STM32G55x/56x products, on S-AHB system bus of the Cortex®-M33 processor to improve the performance of data traffic to/from external memories.

Some specific features, like hit-under-miss and critical-word-first refill policy, optimize performance on external memories data accesses.

9.2 DCACHE main features

The main features of DCACHE are described below:

- Bus interface
  - One 32-bit AHB slave port, the system port (input from Cortex®-M33 S-AHB system interface)
  - One 32-bit AHB master port (output to main AHB bus matrix)
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for access to DCACHE registers)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Optimized cache line refill thanks to WRAP bursts of the size of the cache line (such as WRAP4 for 128-bit cache line)
  - 2-way set-associative
  - Supports both write-back and write-through policies (selectable with AHB bufferable attribute)
  - Read and write-back always allocate
  - Write-through always non-allocate (write-around)
  - Supports byte, half-word, and word writes
- Replacement and refill
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Critical-word-first refill policy for read transactions, minimizing processor stalls
  - Possibility to configure burst type of all AHB memory transactions: INCRw or WRAPw (size w aligned on cache line size)
- ECC hardware support: single-bit error correction, double-bit error detection (SEC-DED)
- Performance counters

  The DCACHE implements four performance counters:
  - Two hit-monitor counters (32-bit): number of read hits, number of write hits
  - Two miss-monitor counters (16-bit): number of read misses, number of write misses

- Error management
  - Possibility to detect error for master port request initiated by DCACHE itself (a cache line written back into main memory, because of an eviction or a clean operation), to flag this error, and optionally to raise an interrupt
- TrustZone security support
- Maintenance operations
  - Cache invalidate: full cache invalidation, fast command, non-interruptible
  - Cache invalidate range: invalidates cache lines whose address belongs to defined range, background task, interruptible
  - Cache clean range: cleans cache lines (if dirty bit = 1, write back line, then clear dirty bit) whose address belongs to defined range, background task, interruptible
  - Cache clean and invalidate range: cleans and invalidates cache lines (if dirty bit = 1, write back line, then invalidate it) whose address belongs to defined range, background task, interruptible

9.3 DCACHE implementation

The DCACHE1 is placed on Cortex®-M33 S-AHB bus, and caches only the external RAM memory region (OCTOSPI), in the address range [0x6000 0000:0x9FFF FFFF] of the memory map.

Indeed, by placing a bus matrix demultiplexing node in front of the DCACHE1, S-AHB bus memory requests addressing SRAM region or peripherals region (respectively in ranges [0x2000 0000:0x3FFF FFFF] and [0x4000 0000:0x5FFF FFFF]) are routed directly to the main AHB bus matrix, and the DCACHE1 is bypassed.

Table 91. DCACHE features

Features	DCACHE1
Number of ways	2
Cache size	4 Kbytes
Cache line width	16 bytes
Data size of AHB Master interface	32 bits
ECC support	Yes

9.4 DCACHE functional description

The purpose of the data cache is to cache external memory data loads and stores, coming from the processor. These accesses include the instruction fetches that may occur at an external memory address. The DCACHE manages both read and write transactions.

The DCACHE also implements the hardware support for ECC, with a single-bit error correction and a double-bit error detection (SEC-DED; see *Section 9.4.9* for more details).

9.4.1 DCACHE block diagram

9.4.2 DCACHE reset and clocks

The DCACHE is clocked on the Cortex®-M33 S-AHB bus clock.

When the DCACHE reset signal is released, a cache invalidate procedure is automatically launched, making the DCACHE busy (DCACHE_SR = 0x0000 0001).

When this procedure is finished:

- The DCACHE is invalidated: cold cache, with all cache line valid, dirty, and privilege bits = 0 and also all TAG addresses, ECC bit-fields and data reset to 0 (DCACHE must be filled up)
- DCACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- The DCACHE is disabled: the EN bit in DCACHE_CR holds its reset state (= 0).

Note: When disabled, the DCACHE is bypassed: slave input requests are just forwarded to the master port.

9.4.3 DCACHE TAG memory

The DCACHE TAG memory contains:

- address tags that indicate which data are contained in the cache data memories
- validity bits
- dirty bits
- privilege bits

There is one valid bit, one dirty bit, and one privilege bit per cache line (per way).

The valid bit enables/disables access to the data cache line: if the line is not valid, the data access (read or write) is performed in the main memory.

The valid bit is set when the cache line is written (refilled by either a read miss or a write-back miss).

Valid bits and also TAG addresses, ECC bit-fields and data are reset in any of the below cases:

- after the DCACHE reset is released
- when the cache is disabled, by setting EN = 0 in DCACHE_CR (by software)
- when executing one of the DCACHE invalidate commands, setting by software CACHEINV = 0, or CACHECMD = 0b010 or 0b011 in DCACHE_CR (see *Section 9.4.8*).

The dirty bit indicates that the cache line has up-to-date values with respect to the main memory content (the cache has last right value, the main memory is not up to date).

The dirty bit is set when the cache line is written by a slave port write transaction (only in case of an access with write-back attribute).

Dirty bits are reset in any of the below cases:

- after the DCACHE reset is released
- when a line refill is performed on a read miss (on a write-back miss, the refilled cache line is modified by the written data, and dirty bit = 1)
- when the cache invalidation is performed
- when executing one of the DCACHE clean operations (cache line written back to the main memory), setting by software CACHECMD = 0b001 or 0b011 in DCACHE_CR (see *Section 9.4.8*).

The privilege bit indicates if the data is managed by a privileged entity. It is assigned according to the value of AHB privileged attribute at the input slave port, for the first access to this line (it is written only during the line refill, on read miss or write-back miss).

The privilege bit holds the same polarity as the privileged attribute: 1 for privileged access, 0 for unprivileged access.

Privilege bits are reset when the cache is invalidated, and after the DCACHE reset is released.

When a cacheable transaction is received at input slave port, its AHB address (HADDR_in) is split into the following fields (see the table below for B and W values):

- HADDR_in[B-1:0]: address byte offset, indicates which byte to select within a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to TAG memory address to check if the requested data is already available (meaning valid) within the DCACHE

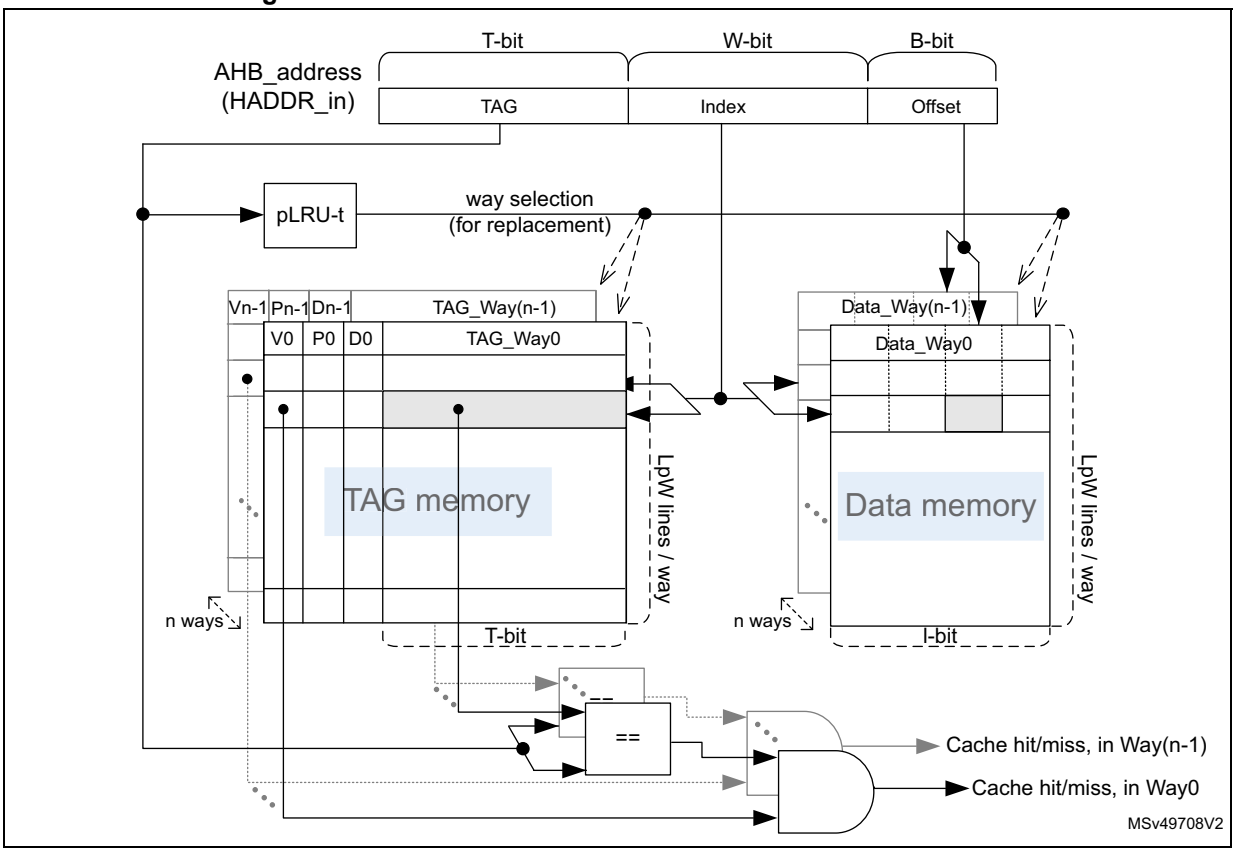
Table 92 gives DCACHE main parameters for TAG memory dimensioning.

Figure 34 shows the functional view of TAG and data memories, for an n-way set associative DCACHE.

Table 92. TAG memory dimensioning parameters

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	4 Kbytes = 4096 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines (per way)	LpW = s / (n x L) lines/way	128 lines/way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	7-bit
TAG address size	T = (32 - W - B) bit	21-bit

Figure 34. DCACHE TAG and data memories functional view



9.4.4 DCACHE enable

To activate the DCACHE functioning, the EN bit must be set in DCACHE_CR control register.

When DCACHE is disabled, it is bypassed and all transactions are copied from slave port to master port in the same clock cycle, and no comparison is performed with TAG address.

DCACHE is by default disabled at boot.

9.4.5 Cacheable and noncacheable traffic

DCACHE is developed for a Cortex®-M33 core and caches the memory regions addressed by the AHB bus connected to it.

In addition, the AHB bus traffic to the memory regions can be cacheable or noncacheable. An incoming memory request to DCACHE is defined as cacheable according to its AHB transaction memory lookup attribute.

In case of write transaction, the DCACHE write policy is defined as write-through or write-back according to its AHB transaction memory bufferable attribute (see *Table 93*).

These AHB attributes depend on the memory protection unit (MPU) programming for the addressed region.

Table 93. DCACHE cacheability for AHB transaction

AHB lookup attribute	AHB bufferable attribute	Cacheability
0	x	Read and write: non cacheable
1	0	Read: cacheable Write: (cacheable) write-through
1	1	Read: cacheable Write: (cacheable) write-back

In case of noncacheable access, the DCACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port.

The bypass does not increase the latency of the access to the targeted memory.

In case of cacheable access, the DCACHE behaves as explained in *Section 9.4.6*.

Cacheable memory regions are defined and programmed by the user in the MPU, responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

9.4.6 Cacheable accesses

When the DCACHE receives a cacheable transaction from Cortex®-M33 on its slave port, the DCACHE checks if the address requested is present in its TAG memory and if the corresponding cache line is valid.

For **read** transaction, there are three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache read hit**, the data is read from cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache read miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first refill policy insures minimum wait cycles for the processor, since read data can be provided while cache is still performing cache line refill (associated latency is the latency of fetching one word from main memory).

  The kind of burst generated on the DCACHE master bus depends on HBURST bit in DCACHE_CR: either INCRw or WRAPw (w being the cache line width, in words).

  The AHB transaction attributes are also propagated from the slave input (missing) request to the master output refill request.
- The address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache read hit** as well (hit-under-miss feature).

  Whatever the line refill is due to a read or write (missing) transaction, the DCACHE can provide the requested read data as soon as the data is available at its master interface, thus avoiding a miss (with data fetch from main memory).

For **write-back** transaction (write transaction, with write-back bufferable attribute), there are three alternatives as well:

- The address is present inside the TAG memory, cache line is valid: **cache write-back hit**, the data is written in cache.
- The address is not present in the TAG memory (or the cache line is not valid): **cache write-back miss**.

  First, a line allocation is performed by reading the entire cache line data from main memory. The kind of burst generated on the DCACHE master bus for this line refill depends on HBURST bit in DCACHE_CR: either INCRw or WRAPw (w being the cache line width, in words), and the AHB transaction attributes are propagated from the slave port initial request.

  Once the refilled line is written in the DCACHE, the initial data provided on the slave port is written in this DCACHE line (it overwrites the data part of the cache line that was refilled just before).
- The address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache write-back hit** as well (hit-under-miss feature).

  Whatever the line refill is due to a read or write (missing) transaction, the DCACHE can write incoming data directly inside the refilled line, thus avoiding a miss (with refill from main memory).

For **write-through** transaction (write transaction, with write-through bufferable attribute), only two alternatives exist:

- The address is present inside the TAG memory, cache line is valid: **cache write-through hit**, the data is written both in cache and in main memory (through master port).
- The address is not present in the TAG memory (or the cache line is not valid): **cache write-through miss**, the data incoming at slave port is written only in main memory (unlike the write-back miss, there is no line allocation and data written in cache).

In case of cache refill (due to cache miss), the DCACHE selects which cache line is written with the refill data: as a 2-way set associative cache, one line among two can be used (line pointed by the address index, in each of the two ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be evicted first:

- If the dirty tag of this line equals 0 (clean data), the line is simply invalidated.
- If it equals 1 (dirty data), the line must be written back in the main memory.

  The DCACHE generates a burst write transaction on its master port, with burst type set to INCRw (w being the cache line width, in words), and with AHB memory transaction attribute signals set as below:
  - data (not instruction)
  - privileged = TAG privilege bit
  - write-back (even if it does not care)
  - normal memory
  - cacheable
  - allocate (even if it does not care)

– nonshareable

These AHB attributes cannot be propagated from the slave port (as it is the case for all other transactions emitted by the DCACHE) because the evicting transaction has no relation with the initial missing transaction. Setting of the AHB attributes is fixed, except for the privileged bit that is copied from the TAG privilege bit of the evicted line.

9.4.7 DCACHE security

The DCACHE implements an Armv8-M TrustZone.

DCACHE configuration registers are protected at system level.

9.4.8 DCACHE maintenance

The DCACHE features several maintenance operations that the software can programmed in DCACHE_CR control register:

- **Full invalidate**: invalidates (and initializes) the whole cache, non interruptible task.

  The software can invalidate the whole DCACHE content by programming CACHEINV in DCACHE_CR.

  When CACHEINV = 1, the DCACHE control logic sets BUSYF flag in DCACHE_SR status register, and performs the operation of cache invalidation, resetting each TAG valid bit to 0 (one valid bit per cache line). Each dirty and privilege bits and TAG addresses, ECC bit-fields and data are also reset to 0 during cache invalidation to prevent unknown values and potential ECC errors detection at next cache line validation. CACHEINV is automatically cleared.

  Once the full invalidate operation is finished, the DCACHE automatically clears BUSYF flag, and sets BSYENDF in DCACHE_SR.

  If enabled on this flag condition (BSYENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised. Then, the (empty) cache is available again.

  This full invalidate operation is not interruptible, meaning that the cache does not treat any cacheable request while BUSYF = 1. However, non-cacheable traffic is treated (since the request address is not compared to TAG ones), the DCACHE being bypassed in the same clock cycle (same behavior as when the DCACHE is disabled).

- **Invalidate range**: invalidates (and initializes) a certain range of addresses in the cache, background task (interruptible).

  The software can invalidate a given data region in the DCACHE by programming STARTCMD = 1 and CACHECMD = 0b010 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  The DCACHE control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), the

corresponding cache line is invalidated (line TAG bits cleared, valid bit = dirty bit = privilege bit = TAG address = ECC = data = 0).

When STARTCMD is set, the DCACHE control logic sets BUSYCMDF in DCACHE_SR and launches the invalidate range operation. STARTCMD is also automatically cleared.

Once the operation is finished (all TAG memory parsed), the DCACHE automatically clears BUSYCMDF and sets CMDENDF in DCACHE_SR.

If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised.

During this invalidate range operation, the DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the invalidation process. The TAG memory is accessed for invalidate range operation only if not already accessed by an external cache request. This implies that invalidate range execution is usually not performed in one go, but can be interrupted.

- **Clean range**: cleans a certain range of addresses in the cache, background task (interruptible).

  Cleaning a cache line means making sure that the main memory content is up-to-date with the data, which may have been modified in the cache. The clean operation consists in performing the write-back in the main memory of the cache lines that are tagged as “dirty” (the ones with TAG dirty bit set).

  The software can clean a given data region in DCACHE by programming STARTCMD = 1, and CACHECMD = 0b001 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  The DCACHE control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), and the corresponding line is dirty, this line is cleaned, meaning the whole cache line is written-back in the memory through the DCACHE master port, and its TAG dirty bit is cleared, and then its ECC bit-field is updated.

  When STARTCMD is set, the DCACHE control logic sets BUSYCMDF in DCACHE_SR and launches the clean range operation. STARTCMD is also automatically cleared.

  Once the operation is finished (all TAG memory parsed), the DCACHE automatically clears BUSYCMDF and sets CMDENDF in DCACHE_SR.

  If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised.

  During this clean range operation, the DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the cleaning process. The TAG memory is accessed for clean range operation only if not already accessed by an external cache request. This implies that clean range execution is usually not performed in one go, but can be interrupted.

  It is under the software responsibility that no bus initiator attempts to change the content of the region being cleaned until clean range is completed. For that, the software can take advantage of BUSYCMDF flag in DCACHE_SR, and can poll this flag to prevent any spurious access to the area being cleaned.

Alternatively it can also rely on the command end flag (CMDENDF) or on the DCACHE interrupt to detect the end of the clean range execution.

- **Clean and invalidate range**: cleans and invalidates (and initializes) a certain range of addresses in the cache, background task (interruptible).

  This operation cleans the “dirty” cache lines that belong to the operation address range (the same as the clean range operation), and also invalidates all the (valid) cache lines that belong to this address range (whether they are dirty or not).

  The software can launch this clean and invalidate range operation, by programming STARTCMD = 1, and CACHECMD = 0b011 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  This sets and clears the same flags, and potentially the same interrupt as invalidate range or clean range operations.

9.4.9 DCACHE ECC support

An Error Correction Code (ECC) hardware is implemented in the DCACHE in order to detect and correct errors when a cache line is read.

This DCACHE ECC hardware enables the detection and the correction of single-bit errors and the detection of double-bit errors, and is then called SEC-DED ECC.

The ECC feature is implemented by the addition (in TAG memory) of an extra ECC bit-field to each DCACHE cache line, in order to provide some redundancy in the TAG + data memories content.

The ECC bit-field size depends on the cache line size (TAG and data).

When the ECC function is enabled (by setting ECCEN = 1 in DCACHE_CR), the ECC bit-field:

- is encoded by the ECC encoder, based on both the TAG (including address and valid bits) and the data to write in a given cache line (when the DCACHE is refilled from the main memory, on cache miss occurrence; or on cache line read-modify-write operation, on a write access that hits)
- is written along with the cache line TAG
- is read along with the TAG of any cache line (when the cache line TAG is accessed to check if its address matches or not with the address of the incoming DCACHE read or write request or the address range of a maintenance range operation).
- is decoded along with the read cache line (TAG and data), to detect if one or two bits are erroneous in this TAG and/or data
  - if the ECC decoder detects a single-bit error, it corrects the erroneous bit (on the fly, by swapping its value between 0 and 1), either in the TAG bit-field (and then it is the corrected TAG which is compared to the address of the incoming request), or in the data bit-field (in order to provide a corrected data, in case of cache hit).

    On a DCACHE read access, the cache line detected with a single-bit error is neither corrected in the cache itself, nor invalidated, and can imply another ECC error detection (and correction) when read on further DCACHE read or write requests.

    On a DCACHE hitting write access, the cache line detected with a single-bit error is not invalidated, and is corrected, when the TAG, the data and the corresponding

ECC bit-field are written back in the cache line (read and correct/modify/write correctly the cache line).

When a cache line refill occurs (because of a cache miss), it may imply a cache line eviction. The cache line candidate for the replacement is read first, to know if it is valid and dirty, or not. If a single-bit error is detected during this read operation, the cache line is corrected (potentially its valid or dirty bit is corrected), and the single-bit error detection is reported. If the valid and dirty bits of the cache line (potentially corrected) are set, its data (potentially corrected as well) is written back to the main memory. If the dirty bit or the valid bit is clear, no further action occurs with the cache line read value (it is simply dropped).

On a DCACHE maintenance range operation (clean and/or invalidate an address range), all cache lines are read to check if their TAG address matches the maintenance address range. If a single-bit error is detected, the TAG or the data is corrected and the maintenance operation is processed correctly (cache line invalidated and/or dirty cache line evicted adequately).

Even if the TAG or the data is corrected on the fly, and then the read or write access, or the eviction or the maintenance operation is safe, the single-bit error detection is reported, to allow the system to measure some DCACHE memories corruption (soft error) and/or aging (see *Section 9.6: DCACHE error management and interrupts* for more details on the related error flags).

– if the ECC decoder detects a double-bit error, it is not able to correct the TAG and/or the data read, and is then unable to decide safely if the request hits or misses (since the TAG may contain errors).

  On read or write-back access to the DCACHE, when such a double-bit error is detected on a cache line in a given cache way, and all other cache lines miss (in the other ways), the read or write operation is aborted, and the DCACHE responds to the incoming request with a bus error.

  For read request, this prevents, in case of a hit (potentially false), the risk of returning a corrupted (and uncorrectable) data from cache; or, in case of a miss (potentially false), the risk of returning and refilling the cache with a data from the main memory which is not necessarily up-to-date (if the first copy of it is marked as dirty in the DCACHE and not yet evicted/written-back in main memory).

  For write-back request, this prevents, in case of a hit (potentially false), the risk of corrupting a cache line (by either writing to a wrong cache line, or by modifying and writing in the cache line a data that were erroneously read); or, in case of a write-back miss (potentially false), the risk of refilling the cache with a data from the main memory which is not necessarily up-to-date.

  On write-through access to the DCACHE, when such a double-bit error is detected on a cache line in a given cache way, and all other cache lines miss (in the other ways), the DCACHE considers the write-through cache access as a miss: the data is written only in the main memory, and is not written into the cache. Compared to a usual (error-less) write-through miss, the only difference is the setting of the ECC double-bit error flag.

On a cache line replacement, the cache line is read first, potentially to perform an eviction (in case this cache line is both valid and dirty), before it is replaced. If a double-bit error is detected on the cache line read, the potential eviction is aborted (the valid and dirty bits may be erroneous, and/or the data written-back to the main memory would be potentially erroneous as well) and the cache line is lost (since replaced/over-written by the new TAG and data from main memory).

On a DCACHE maintenance range operation (clean and/or invalidate an address range), all cache lines are read to check if their TAG address matches the maintenance address range. If a double-bit error is detected on a given cache line read, the maintenance operation is not performed on this cache line, since the double-bit error can affect its valid, dirty or address TAG. This erroneous cache line is not invalidated and/or not copied back to the main memory; and the maintenance operation continues on the other cache lines.

In both cases above, as the potentially needed eviction of a dirty data is not performed, some data may be lost (dirty data were only in cache) and/or the main memory may be not up-to-date, as it was intended.

The cache line detected with a double-bit error is not automatically invalidated in the DCACHE, nor necessarily over-written by a refill operation: then, as long as it remains in the DCACHE, this erroneous cache line may imply another ECC double-bit error detection, when it is read on further DCACHE accesses.

In all cases above, the double-bit error detection is reported (see *Section 9.6: DCACHE error management and interrupts*), in order the system to react and potentially to reset the complete DCACHE content (by disabling the DCACHE, or by launching a full invalidate maintenance operation), considering that some data may be lost and/or the main memory may not be up-to-date. At least this is the only safe way to invalidate such erroneous cache lines (with a double-bit error).

As the TAG (and data) are read, in parallel, in all the ways of the DCACHE, an ECC decoder is implemented for each way of the DCACHE.

Table 94 summarizes the behavior of the DCACHE when the ECC feature is enabled.

Table 94. DCACHE accesses management (when ECC is enabled)

Access type	ECC error detected on cache line (TAG address/valid/dirty bits or data)	Decision and operation	ECC error reporting
Read/ Write-back/ Write-through	No ECC error	Hit or Miss	No
	Single-bit ECC error	Hit or Miss (error corrected on the fly)	ECC single-bit error flag
Read/ Write-back	Double-bit ECC error (and all other ways miss)	Bus error (erroneous cache line still in cache; no data read / data to write in cache lost)	ECC double-bit error flag
Write-through		Miss (erroneous cache line still in cache; main memory up-to-date)	ECC double-bit error flag

Table 94. DCACHE accesses management (when ECC is enabled) (continued)

Access type	ECC error detected on cache line (TAG address/valid/dirty bits or data)	Decision and operation	ECC error reporting
Eviction (replacement)	No ECC error	Eviction	No
	Single-bit ECC error	Eviction (data corrected on the fly)	ECC single-bit error flag
	Double-bit ECC error	Eviction aborted (dirty data lost)	ECC double-bit error flag
Clean and/or invalidate range maintenance operation	No ECC error	Maintenance operation	No
	Single-bit ECC error	Maintenance operation (error corrected on the fly)	ECC single-bit error flag
	Double-bit ECC error	Operation aborted only on the erroneous cache line (erroneous cache line still in cache and/or main memory not up-to-date)	ECC double-bit error flag

When the ECC function is disabled (by setting ECCEN = 0 in DCACHE_CR), the ECC bit-field is neither encoded in the cache lines (not written), nor decoded on the cache lines read, and the DCACHE behaves as if the ECC feature is not implemented (including the ECC single-bit error and double-bit error flags remain clear).

9.4.10 DCACHE performance monitoring

The DCACHE provides the following monitors for performance analysis:

- The two 32-bit read-hit and write-hit monitors count the AHB transactions at the DCACHE input (slave port) that do not generate a transaction on the DCACHE output (master port).

  These monitors also take into account all accesses whose address is present in the TAG memory, or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 9.4.6*).
- The two 16-bit read-miss and write-miss monitors count the AHB transactions at the DCACHE input (slave port) that generate a transaction on the DCACHE output (master port).

  These monitors also take into account all accesses whose address is not present neither in the TAG memory, nor in the refill buffer.

Upon reaching their maximum values, the monitors do not wrap over.

The software can perform the following tasks:

- Enable/stop the read (write) hit monitor, through R(W)HITMEN in DCACHE_CR.
- Reset the read (write) hit monitor, by setting R(W)HITMRST in DCACHE_CR.
- Enable/stop the read (write) miss monitor, through R(W)MISSMEN in DCACHE_CR.
- Reset the read (write) miss monitor, by setting R(W)MISSMRST in DCACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

9.4.11 DCACHE boot

The DCACHE is disabled (EN = 0 in DCACHE_CR) at boot.

Once the boot is finished, the DCACHE can be enabled (software setting EN = 1 in DCACHE_CR).

9.5 DCACHE low-power modes

At product level, the DCACHE reduces the power consumption by loading/storing data from/to the internal DCACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even much higher, since the cached main memories are external.

9.6 DCACHE error management and interrupts

A transaction initiated on the DCACHE master port may return an error (a write attempt into a read-only memory, for instance). If the master port request was propagated from a slave port request, the error is propagated back to the slave port. If ever the master port request is initiated by the DCACHE itself (a cache line is written back into the main memory because of an eviction or a clean operation), the DCACHE receives this functional error and flags it internally by setting the ERRF flag in DCACHE_SR.

In such a case, an interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in DCACHE_IER).

When the ECC function is enabled (by setting ECCEN = 1 in DCACHE_CR), two ECC error flags can be set in DCACHE_SR:

- the ECC single-bit error flag, SERRF.
- the ECC double-bit error flag, DERRF.

and an interrupt is then generated if the corresponding interrupt enable bit is set (SERRIE = 1, respectively DERRIE = 1, in DCACHE_IER).

The detection of an ECC double-bit error also drives a related DCACHE double-bit error signal, in order to potentially launch a timer break (the double-bit error signal level is a copy of the DERRF flag value).

Another case of potential interrupt generation is at the end of a full invalidate operation: when the cache busy state is finished, the DCACHE sets BSYENDF flag in DCACHE_SR.

An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in DCACHE_IER).

Last case is at the end of invalidate and/or clean range operations: when the command busy state is finished, the DCACHE sets CMDENDF flag in DCACHE_SR.

An interrupt is also generated if the corresponding interrupt enable bit is set (CMDENDIE = 1 in DCACHE_IER).

All DCACHE interrupt sources raise the same and unique interrupt signal, dcache_it, and then use the same interrupt vector.

Table 95. DCACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
DCACHE	Functional error	ERRF in DCACHE_SR	ERRIE in DCACHE_IER	Set CERRF to 1 in DCACHE_FCR
	ECC single-bit error	SERRF in DCACHE_SR	SERRIE in DCACHE_IER	Set CSERRF to 1 in DCACHE_FCR
	ECC double-bit error	DERRF in DCACHE_SR	DERRIE in DCACHE_IER	Set CDERRF to 1 in DCACHE_FCR
	End of busy state (full invalidate finished)	BSYENDF in DCACHE_SR	BSYENDIE in DCACHE_IER	Set CBSYENDF to 1 in DCACHE_FCR
	End of cache operations (address range based)	CMDENDF in DCACHE_SR	CMDENDIE in DCACHE_IER	Set CCMDENDF to 1 in DCACHE_FCR

The DCACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from master port back to the S-AHB slave port.

9.7 DCACHE registers

9.7.1 DCACHE control register (DCACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	STARTCMD	CACHECMD[2:0]			Res.	Res.	Res.	ECCEN	Res.	Res.	CACHEINV	EN
				w	rw	rw	rw				rw			w	rw

Bit 31 **HBURST:** output burst type for cache master port read accesses
Can be set by software, only when EN = 0.
Master port write accesses are always done in INCR burst type.
0: WRAP
1: INCR

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **WMISSMRST:** write-miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache write-miss monitor

Bit 22 **WHITMRST:** write-hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache write-hit monitor

Bit 21 **WMISSMEN:** write-miss monitor enable
0: cache write-miss monitor switched off. Stopping the monitor does not reset it.
1: cache write-miss monitor enabled

Bit 20 **WHITMEN:** write-hit monitor enable
0: cache write-hit monitor switched off. Stopping the monitor does not reset it.
1: cache write-hit monitor enabled

Bit 19 **RMISSMRST:** read-miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache read-miss monitor

Bit 18 **RHITMRST:** read-hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache read-hit monitor

Bit 17 **RMISSMEN:** read-miss monitor enable
0: cache read-miss monitor switched off. Stopping the monitor does not reset it.
1: cache read-miss monitor enabled

Bit 16 **RHITMEN:** read-hit monitor enable
0: cache read-hit monitor switched off. Stopping the monitor does not reset it.
1: cache read-hit monitor enabled

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **STARTCMD:** starts maintenance command (maintenance operation defined in CACHECMD).
Can be set by software, only when EN = 1, BUSYCMDF = 0, BUSYF = 0 and CACHECMD = 0b001, 0b010 or 0b011.
Cleared by hardware when the BUSYCMDF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: command operation (cache maintenance) finished
1: start maintenance command (cache maintenance)

Bits 10:8 **CACHECMD[2:0]:** cache command maintenance operation (cleans and/or invalidates an address range)
Can be set and cleared by software, only when no maintenance command is ongoing (BUSYCMDF = 0).
000: no operation
001: clean range
010: invalidate range
011: clean and invalidate range
others: reserved

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ECCEN**: ECC enable
ECCEN can be modified (by software) only when the cache is disabled (EN = 0).
0: ECC encoder and decoder disabled
1: ECC encoder and decoder enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **CACHEINV:** full cache invalidation
Can be set by software, only when EN = 1.
Cleared by hardware when the BUSYF flag is set (during full cache invalidation operation).
Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit and TAG/ECC/data = 0)

Bit 0 **EN:** enable
0: cache disabled
1: cache enabled

9.7.2 DCACHE status register (DCACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DERRF	SERRF	Res.	Res.	Res.	CMDENDF	BUSYCMDF	ERRF	BSYENDF	BUSYF
						rw	rw				r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **DERRF**: ECC double-bit error flag
DERRF can be set to 1 by software for debug purpose; writing 0 has no effect (clearing through CDERRF in DCACHE_FCR)
0: no ECC double-bit error detected
1: an ECC double-bit error was detected

Bit 8 **SERRF**: ECC single-bit error flag
SERRF can be set to 1 by software for debug purpose; writing 0 has no effect (clearing through CSERRF in DCACHE_FCR)
0: no ECC single-bit error detected
1: an ECC single-bit error was detected

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CMDENDF:** command end flag
Cleared by writing DCACHE_FCR.CCMDENDF = 1.
0: cache busy or in idle
1: CACHECMD command finished

Bit 3 **BUSYCMDF:** command busy flag
0: cache not busy on a CACHECMD command
1: cache busy on a CACHECMD command (clean and/or invalidate an address range)

Bit 2 **ERRF:** cache error flag
Cleared by writing DCACHE_FCR.CERRF = 1.
0: no error
1: an error occurred during the operation (eviction or clean operation write-back error).

Bit 1 **BSYENDF:** full invalidate busy end flag
Cleared by writing DCACHE_FCR.CBSYENDF = 1.
0: cache busy or in idle
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF:** full invalidate busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

9.7.3 DCACHE interrupt enable register (DCACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DERRIE	SERRIE	Res.	Res.	Res.	CMDENDIE	Res.	ERRIE	BSYENDIE	Res.
						rw	rw				rw		rw	rw	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **DERRIE**: interrupt enable on ECC double-bit error detection
Set by software to enable an interrupt generation in case of ECC double-bit error detection
0: interrupt disabled on ECC double-bit error detection
1: interrupt enabled on ECC double-bit error detection

Bit 8 **SERRIE**: interrupt enable on ECC single-bit error detection
Set by software to enable an interrupt generation in case of ECC single-bit error detection
0: interrupt disabled on ECC single-bit error detection
1: interrupt enabled on ECC single-bit error detection

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CMDENDIE**: interrupt enable on command end
Set by software to enable an interrupt generation at the end of a cache command (clean and/or invalidate an address range)
0: interrupt disabled on command end
1: interrupt enabled on command end

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ERRIE:** interrupt enable on cache error
Set by software to enable an interrupt generation in case of cache functional error (eviction or clean operation write-back error)
0: interrupt disabled on error
1: interrupt enabled on error

Bit 1 **BSYENDIE:** interrupt enable on busy end
Set by SW to enable an interrupt generation at the end of a cache full invalidate operation.
0: Interrupt disabled on busy end
1: Interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

9.7.4 DCACHE flag clear register (DCACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CDERRF	CSERRF	Res.	Res.	Res.	CCMDENDF	Res.	CERRF	CBSYENDF	Res.
						w	w				w		w	w	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CDERRF**: clear ECC double-bit error flag
Set by software.
0: no effect
1: clears DERRF flag in DCACHE_SR

Bit 8 **CSERRF**: clear ECC single-bit error flag
Set by software.
0: no effect
1: clears SERRF flag in DCACHE_SR

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CCMDENDF:** clear command end flag
Set by software.
0: no effect
1: clears CMDENDF flag in DCACHE_SR

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CERRF:** clear cache error flag
Set by software.
0: no effect
1: clears ERRF flag in DCACHE_SR

Bit 1 **CBSYENDF:** clear full invalidate busy end flag
Set by software.
0: no effect
1: clears BSYENDF flag in DCACHE_SR

Bit 0 Reserved, must be kept at reset value.

9.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RHITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RHITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RHITMON[31:0]:** cache read-hit monitor counter

9.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RMISSMON[15:0]:** cache read-miss monitor counter

9.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WHITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WHITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WHITMON[31:0]:** cache write-hit monitor counter

9.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WMISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WMISSMON[15:0]:** cache write-miss monitor counter

9.7.9 DCACHE command range start address register (DCACHE_CMDRSADDRR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDSTARTADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDSTARTADDR[15:4]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:4 **CMDSTARTADDR[31:4]:** start address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies

This register must be set before DCACHE_CR.CACHECMD is written.

Bits 3:0 Reserved, must be kept at reset value.

9.7.10 DCACHE command range end address register (DCACHE_CMDREADDRR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDENDADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDENDADDR[15:4]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:4 **CMDENDADDR[31:4]:** end address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies

This register must be set before DCACHE_CR.CACHECMD is written.

Bits 3:0 Reserved, must be kept at reset value.

9.7.11 DCACHE register map

Table 96. DCACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DCACHE_CR	HBURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN	Res.	Res.	Res.	Res.	STARTCMD	CACHECMD[2:0]			Res.	Res.	Res.	ECCEN	Res.	Res.	CACHEINV	EN
	Reset value	0								0	0	0	0	0	0	0	0					0	0	0	0				0			0	0
0x004	DCACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERRF	SERRF	Res.	Res.	Res.	CMDENDF	BUSYCMDF	ERRF	BSYENDF	BUSYF
	Reset value																							0	0				0	0	0	0	1
0x008	DCACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERRIE	SERRIE	Res.	Res.	Res.	CMDENDIE	Res.	ERRIE	BSYENDIE	Res.
	Reset value																							0	0				0		0	0	
0x00C	DCACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDERRF	CSERRF	Res.	Res.	Res.	CCMDENDF	Res.	CERRF	CBSYENDF	Res.
	Reset value																							0	0				0		0	0	
0x010	DCACHE_RHMONR	RHITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DCACHE_RMMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DCACHE_WHMONR	WHITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DCACHE_WMMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DCACHE_CMDRSADDRR	CMDSTARTADDR[31:4]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x02C	DCACHE_CMDREADDRR	CMDENDADDR[31:4]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

10 Power control (PWR)

10.1 Introduction

The power controller manages the device power supplies and power modes transitions.

10.2 PWR main features

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain (V_{CORE})
  - V_{DD} domain
  - RTC domain
  - Analog domain (V_{DDA})
  - Supply for the SMPS power stage (available on SMPS packages)
  - V_{DD} for USB transceiver
- System supply voltage regulation
  - SMPS step-down converter
  - Linear voltage regulator (LDO)
  - V_{CORE} provided from an external supply (bypass)
- Power supply supervision
  - POR/PDR monitoring
  - PVD monitoring
  - AVD monitoring
  - Out of functional range temperature monitoring
  - Maximum V_{DD} voltage monitoring
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- TrustZone® security and privileged protection

10.3 PWR pins and internal signals

Table 97. PWR input/output pins

Pin name	Signal type	Description
VDD	Supply	Main supply
GND	Supply	Main ground
VDDA	Supply	Analog peripherals supply
VSSA	Supply	Analog peripherals ground
VCAP	Supply	Logic supply (V_{CORE})
VDDSMPS	Supply	SMPS supply
VSSSMPS	Supply	SMPS ground
VLXSMPS	Supply	SMPS output
VREF+	Supply	ADC high reference voltage
VREF-	Supply	ADC low reference voltage
VREFDAC+	Supply	DAC high reference voltage
VREFDAC-	Supply	DAC low reference voltage
WKUPx (x = 1, 2, 4 to 8)	Input	Wake-up pins
CSLEEP	Output	MCU in Sleep mode
CSTOP	Output	CPU in Stop modes

Table 98. PWR internal input/output signals

Internal signal name	Signal type	Description
WKUPx (x = 1, 2, 4 to 8)	Input	Wake-up event source

10.4 PWR power supplies and supply domains

Figure 35. Power supply with SMPS

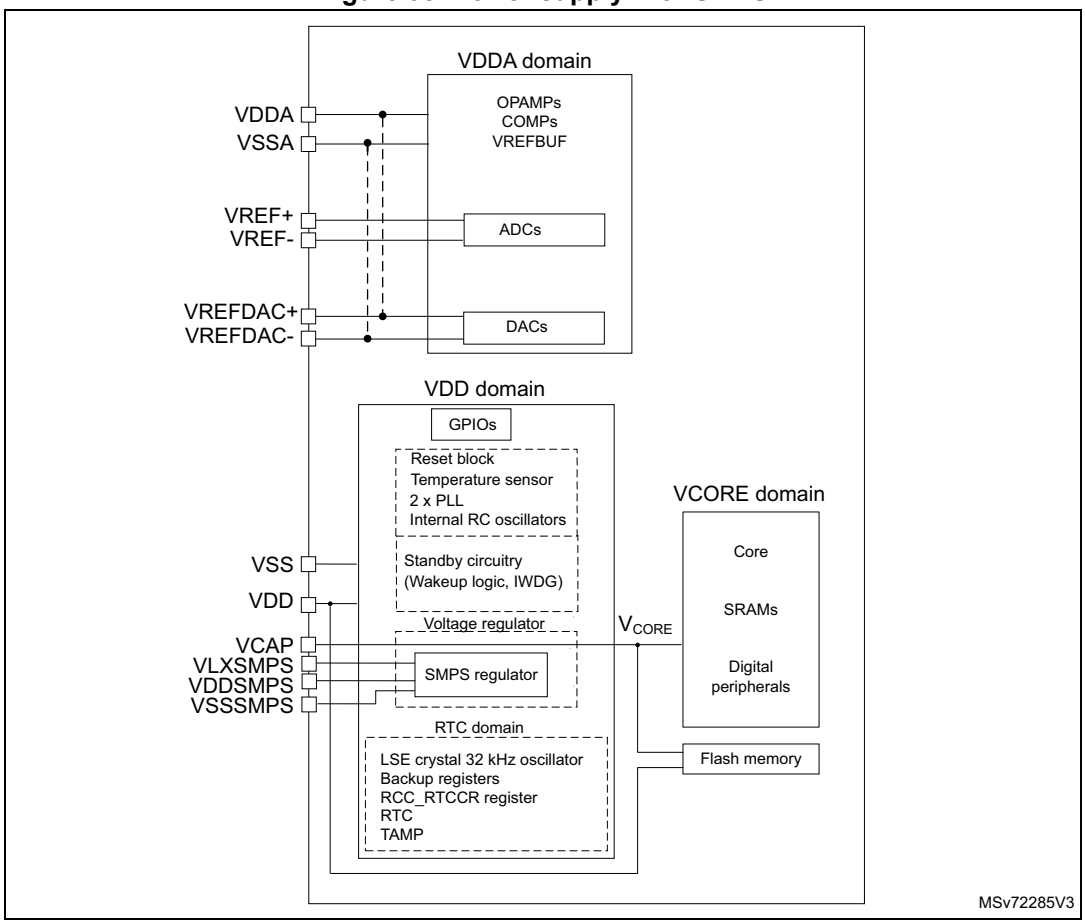
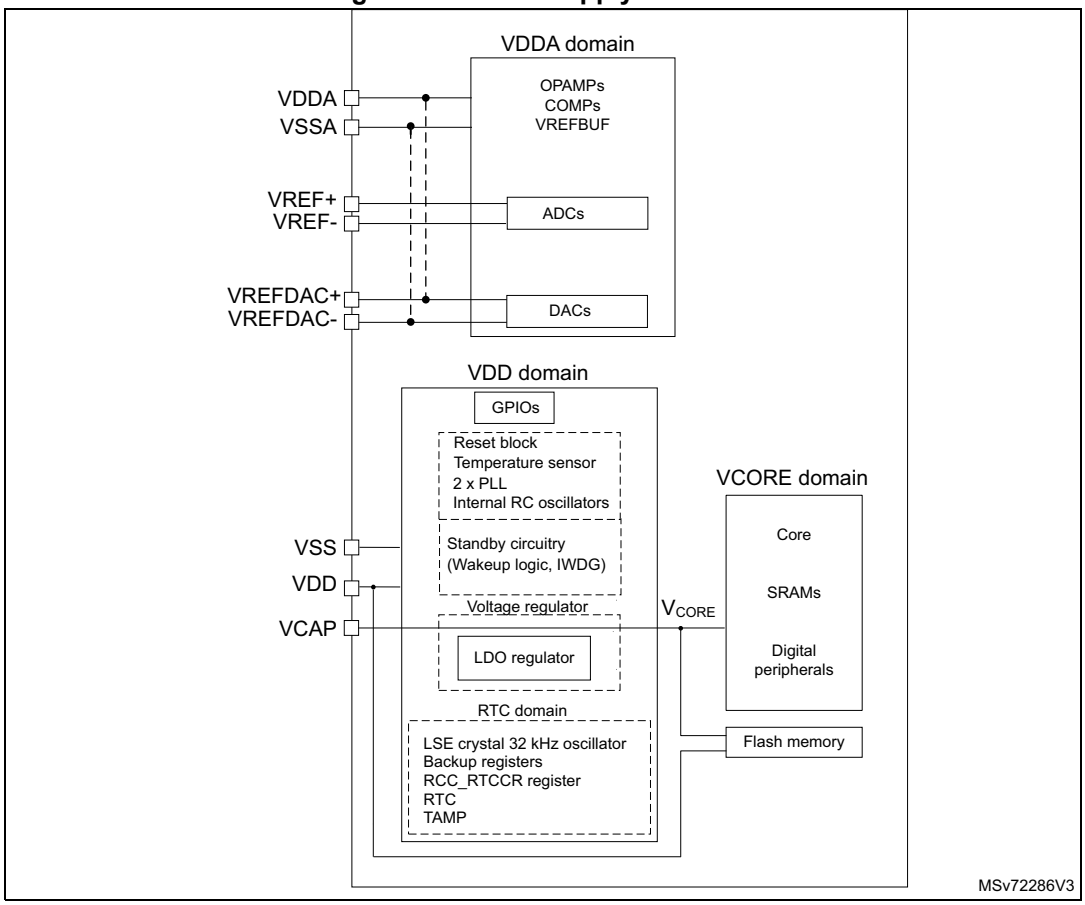


Figure 36. Power supply with LDO



10.4.1 External power supplies

The devices require a 2.7 V to 3.6 V V_{DD} operating voltage supply. Several independent supplies can be provided for specific peripherals:

- V_{DD} = 2.7 V to 3.6 V

  V_{DD} is the external power supply for the I/Os, the internal regulator, and the system analog such as reset, power management, and internal clocks. It is provided externally through the VDD pins.

  V_{DD} is the power supply for RTC, LSE, external clock 32 kHz oscillator, and backup registers.

- V_{DDA} = 2.7 V to 3.6 V

  V_{DDA}/V_{SSA} is the external analog power supply for ADCs, DACs, COMPs, OPAMPs, and VREFBUF. The V_{DDA} voltage level is independent from the V_{DD} voltage and must preferably be connected to VDD when these peripherals are not used.

  If VDDA/VSSA pins are not available on the package, then these pins are double bonded to VDD/VSS.

- V_{DDSMPS} = 2.7 V to 3.6 V

  V_{DDSMPS}/V_{SSSMPS} is the external power supply for the SMPS step-down converter. It is provided externally through VDDSMPS supply pin and must be connected to the same supply as VDD.

- V_{LXSMPS} is the switched SMPS step-down converter output. The SMPS power supply pins are available only on packages with SMPS step-down converter option.

- V_{REF+}/V_{REF-}

  V_{REF+}/V_{REF-} is the input reference voltage for ADCs. It is also the output of the internal voltage reference buffer when enabled (VREFBUF).

  VREF+ can be grounded when ADCs are not active.

  VREF+ and VREF-pins are not available on all packages. When not available, they are bonded to VDDA and VSSA, respectively.

  When the VREF+ is double-bonded to VDDA in a package, the internal voltage reference buffer is not available and must be kept disabled.

  V_{REF-} must always be equal to V_{SSA}.

- $V_{REFDAC+}/V_{REFDAC-}$

  $V_{REFDAC+}/V_{REFDAC-}$is the input reference voltage for DACs.

  VREFDAC+ can be grounded when DACs are not active.

  VREFDAC+ and VREFDAC- pins are not available on all packages. When not available, they are bonded to VDDA and VSSA, respectively.

  $V_{REFDAC-}$ must always be equal to V_{SSA}.

10.4.2 Internal and external regulators

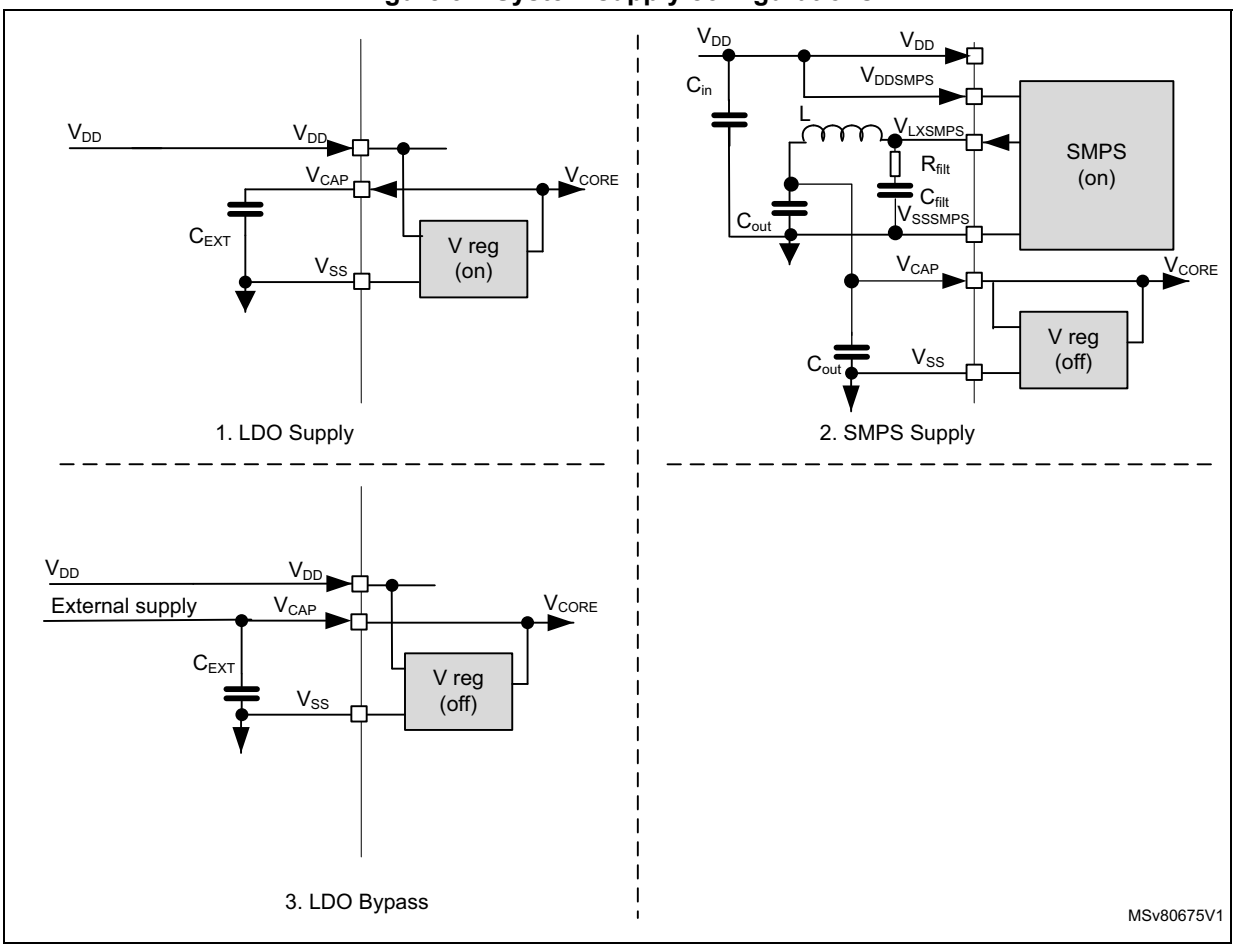
Depending upon the package, the devices embed an LDO and/or an SMPS regulator, to provide the V_{CORE} supply for digital peripherals, SRAM1, SRAM2, SRAM3, and embedded flash memory. The SMPS generates this voltage on VCAP pins, with an external capacitor of 10 µF (typical) on each VCAP pin. The SMPS requires an external coil. The LDO generates this voltage on VCAP pins connected to external capacitors of 2.2 µF each (typical). If only one VCAP pin is present on the package, the sum of the capacity is used for this one VCAP pin (4.7 uF for LDO devices). The SMPS requires an external capacitor 220 pF in

series with a resistor of 18 Ω between VLXSMPS and VSSSMPS pins. For LDO devices, it is not necessary to connect all VCAP pins together (it is optional). For SMPS devices, all VCAP pins must be connected together with low impedance path.

Both regulators can provide three different voltages (voltage scaling), and can operate in Stop mode.

Depending on the package configuration (SMPS or LDO), the regulator is selected by hardware. SMPS and LDO regulator are exclusively selected. The supply configurations shown in *Figure 37* are supported for the VCORE domain supply.

Figure 37. System supply configurations

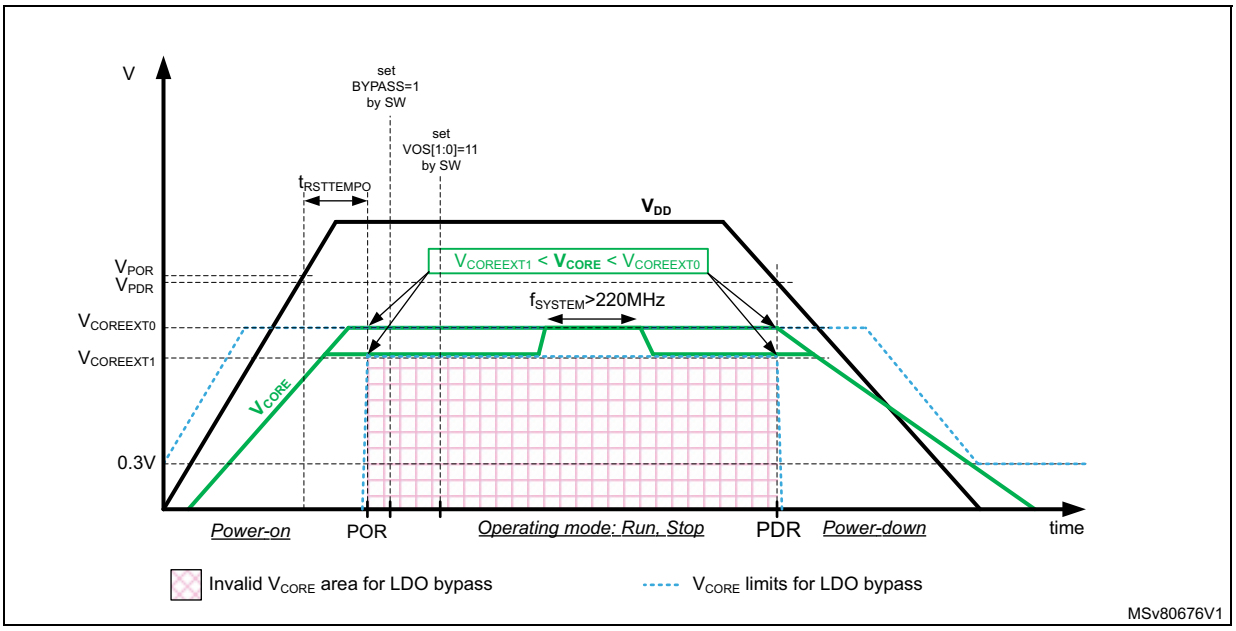


Startup with V_{CORE} provided from an external supply (bypass)

In addition to LDO or SMPS regulators that provide the V_{CORE} supply, the devices support LDO bypass mode on all non-SMPS packages. In bypass mode, the V_{CORE} supply is provided from the external power supply to VCAP pins. In this configuration, the device is started in standard LDO mode but the external power supply must provide V_{CORE} voltage to VCAP pins at a level always higher than the output from the internal LDO regulator. The default V_{CORE} startup range is VOS2, therefore the external voltage during startup must be at higher $V_{COREEXT1}$ or $V_{COREEXT0}$ levels (refer to the datasheet values). The external V_{CORE} supply voltage must reach this correct level while V_{DD} reaches the V_{POR} threshold

level + $t_{RSTTEMPO}$ temporization time. V_{CORE} voltage must be always lower than V_{DD} voltage (it is recommended to put a Schottky diode between VCAP and VDD pins to avoid current injection from V_{CORE} to V_{DD}). The firmware must then disable the internal LDO regulator by setting the BYPASS bit in the PWR_SCCR register, and continue to supply V_{CORE} only from the external power supply. The external power supply level must follow V_{CORE} voltage requirements for the selected operation frequency ranges: $V_{COREEXT1}$ (for f_{HCLK} < 220 MHz, T_{jmax} = 140°C) or $V_{COREEXT0}$ (for f_{HCLK} < 280 MHz, T_{jmax} = 105°C. VOS[1:0] bits (in PWR_VOSCR register) must be set to VOS0 range (VOS[1:0] = 3) after setting BYPASS = 1 and before increasing the system frequency above 150 MHz. This VOS0 setting must be used for any operating conditions in bypass mode. Standby mode must not be used in bypass mode. The lowest supported power mode in bypass mode is Stop mode.

Figure 38. Startup sequence if V_{CORE} provided from external supply (LDO bypass)



In LDO bypass mode (BYPASS = 1), the states of ACTVOS[1:0], ACTVOSRDY and VOSRDY bits do not reflect the V_{CORE} voltage range. ACTVOS[1:0], ACTVOSRDY and VOSRDY bit states are valid only if the internal LDO or SMPS regulator is active (BYPASS = 0). The SRAMLS bit reflects the SRAM timing setting in all LDO, SMPS, and Bypass modes (see *Section 10.11.4: PWR voltage scaling status register (PWR_VOSSR)*).

10.4.3 Power-up and power-down power sequences

During power-up and power-down phases, the following power sequence requirements must be respected:

- When V_{DD} is below 1 V, other power supplies (V_{DDA}, V_{REF+}, $V_{REFDAC+}$) must remain below V_{DD} + 300 mV.
- When V_{DD} is above 1 V, all power supplies are independent (but V_{REF+} and $V_{REFDAC+}$ must be below V_{DDA} + 300 mV).

During the power-down phase, V_{DD} can temporarily become lower than other supplies only if the energy provided to the MCU remains below 1 mJ. This allows external decoupling

capacitors to be discharged with different time constants during the power-down transient phase.

10.4.4 Independent analog peripherals supply

To improve analog peripheral conversion accuracy (ADC, DAC, OPAMP, COMP, VREFBUF) and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB:

- The voltage supply input of the analog peripherals is available on a separate VDDA pin.
- An isolated supply ground connection is provided on the VSSA pin.

The V_{DDA} supply voltage can be different from V_{DD}. The presence of V_{DDA} must be checked before enabling any of the analog peripherals supplied by V_{DDA} (ADC, DAC, OPAMP, COMP, VREFBUF).

Power supply level monitoring is available on VDDA via AVDO bit in the PWR_VMSR register.

When a single supply is used, V_{DDA} can be externally connected to V_{DD} through the external filtering circuit to ensure a noise-free V_{DDA} reference voltage.

ADC reference voltage

To ensure a better ADC accuracy, the user can connect to VREF+, a separate reference voltage lower than V_{DDA}. V_{REF+} is the highest voltage represented by the full-scale value for an analog input (ADC).

V_{REF+} can be provided either by an external reference or by an internal buffered voltage reference (VREFBUF). The internal voltage reference (VREFBUF) can output a configurable voltage, namely 1.8 V, 2.048 V, 2.5 V, and 2.9 V. VREFBUF provides the voltage to external components through VREF+ pin. Refer to the device datasheet and to *Section 29: Voltage reference buffer (VREFBUF)* for further information.

VREF+/VREF- pins must be always correctly decoupled by external capacitors. The recommended value for these capacitors is a 1 µF capacitor in parallel with a 100 nF capacitor. The capacitors must be placed very close to VREF+ and VREF- pins. Correct decoupling of VREF+/VREF- pins has a significant impact on ADC performance. It is recommended to use low ESR capacitors and minimize parasitic inductance from capacitors to VREF+/VREF- pins (by placing capacitors immediately to VREF+/VREF- pins with no vias in the path).

Note: The VREF- pin is not available on all packages (internally connected to VSS).

Do not enable the internal voltage reference buffer when an external power supply is applied to the VREF+ pin.

DAC reference voltage

To ensure a better DAC accuracy, the user can connect to VREFDAC+/VREFDAC- a separate reference voltage lower than V_{DDA}. $V_{REFDAC+}$ is the highest voltage represented by the full-scale value for an analog output signal from a DAC.

$V_{REFDAC+}$ can be provided only by an external reference (or by the external connection of VREFDAC+/VREFDAC- pins to VREF+/VREF- pins).

VREFDAC+/VREFDAC- pins must be always correctly decoupled by external capacitors. The recommended value for these capacitors is a 1 µF capacitor in parallel with a 100 nF

capacitor. The capacitors must be placed very close to VREFDAC+ and VREFDAC- pins. Correct decoupling of VREFDAC+/VREFDAC- pins has a significant impact on DAC performance. It is recommended to use low ESR capacitors and minimize parasitic inductance from capacitors to VREFDAC+/VREFDAC- pins (by placing capacitors immediately to VREFDAC+/VREFDAC- pins with no vias in path).

Note: The VREFDAC+ and VREFDAC- pins are not available on all packages (internally connected, respectively, to VDDA and VSSA).

10.4.5 USB transceivers supply

The USB transceivers are supplied from VDD/VSS pins. For correct USB operation, V_{DD} must range from 3.0 to 3.6 V.

To use the USB peripheral, the USB33SV bit of the PWR_USBSCR register must be set. It is used to validate the V_{DD} supply for electrical and logical isolation purposes.

10.4.6 RTC domain

The RTC domain is powered from the V_{DD} supply. It contains the RTC unit, the backup registers, the TAMP peripheral, the LSE oscillator (PC14/PC15 pins, and PC13 pin), allowing those blocks to operate even in Standby mode (when the V_{CORE} supply domain is switched off). The RTC domain is not reset by system reset.

When the RTC domain operates in Stop or Standby mode, the following pins are available:

- PC13, PC14 and PC15, which can be configured by RTC or LSE (refer to *Section 49.3: RTC functional description*)
- PC13, PA0, and PA1 when they are configured by TAMP peripheral as tamper pins.

RTC domain access

After a system reset, the RTC domain (RCC RTC domain control register RCC_RTCCR, RTC registers, TAMP registers and backup registers) is protected against possible unwanted write accesses. To enable accesses to the RTC domain, set the DRTCP bit in PWR_RTCCR.

10.5 PWR system supply voltage regulation

10.5.1 SMPS and LDO embedded regulators

The devices embed two internal regulators, exclusively enabled by hardware on power-on reset, depending upon package configuration. To supply the V_{CORE} from an external source, it is possible to disable the regulator by setting the BYPASS bit in the PWR_SCCR register.

The BYPASS bit is written once after power-on reset. Written-once mechanism locks the register and any further write access is ignored. The system must be power cycled before writing a new value.

When V_{CORE} is supplied from an external source, refer to *Section : Startup with VCORE provided from an external supply (bypass)*.

Both regulators can provide three different voltages (voltage scaling) and can operate in Stop mode.

10.5.2 V_{CORE} supply versus reset, voltage scaling, and low-power modes

After reset, the V_{CORE} is in the VOS2 range.

When exiting the Stop or Standby mode, the voltage range is the VOS2.

10.5.3 Embedded voltage regulator operating modes

There are three different power modes: Run, Stop, and Standby modes.

Run mode

The voltage regulator provides full power to the V_{CORE} domain (core, memories, and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS0, VOS1, and VOS2) that are configured through the VOS bits in the PWR voltage scaling control register (PWR_VOSCR).

The VOS voltage scaling allows optimization of the power consumption when the system is clocked below the maximum frequency. By default, VOS2 is selected after system reset.

VOSx bits can be changed on-the-fly to adapt to the required system performance.

Stop mode

The voltage regulator supplies the V_{CORE} domain to retain the content of registers and internal memories. The regulator mode is automatically set to the Stop mode range to reduce power consumption (lower voltage level than VOS2).

Standby mode

The regulator is OFF and the V_{CORE} domain is powered down. The content of the registers and memories is lost except for the Standby circuitry and the RTC domain.

The Standby mode must not be used if V_{CORE} is provided from an external supply (bypass).

10.5.4 Regulator bypass operating modes

This mode is available only for packages without an SMPS regulator option.

The external power supply provides the voltage for V_{CORE}. The internal LDO regulator must be disabled by software (by setting the BYPASS bit of the PWR_SCCR register).

Two different power modes can be used: Run and Stop. Refer to section *Startup with VCORE provided from an external supply (bypass)*.

Run mode

The external voltage regulator provides full power to the V_{CORE} domain (core, memories, and digital peripherals).

After the power-on reset, the external regulator must provide a voltage at given range: $V_{COREEXT1}$ (for f_{HCLK} < 220 MHz, T_{jmax} = 140°C) or $V_{COREEXT0}$ (for f_{HCLK} < 280 MHz, T_{jmax} = 105°C).

VOS[1:0] bits (in PWR_VOSCR register) must be set to VOS0 range (VOS[1:0] = 3) after setting BYPASS = 1 and before increasing the system frequency above 150 MHz. This VOS0 setting must be used for any operating conditions in bypass mode.

Refer to *Startup with VCORE provided from an external supply (bypass)*for the requirements on external regulator value, startup/shutdown timings, and level changing sequence.

Stop mode

The external voltage regulator supplies the V_{CORE} domain to retain the content of registers and internal memories. The voltage supplied by the external regulator must be within the range $V_{COREEXT1}$ or $V_{COREEXT2}$.

Standby mode

The Standby mode must not be used if V_{CORE} is provided from an external supply.

10.6 PWR power supply and temperature supervision

Power supply level monitoring is available on the following supplies:

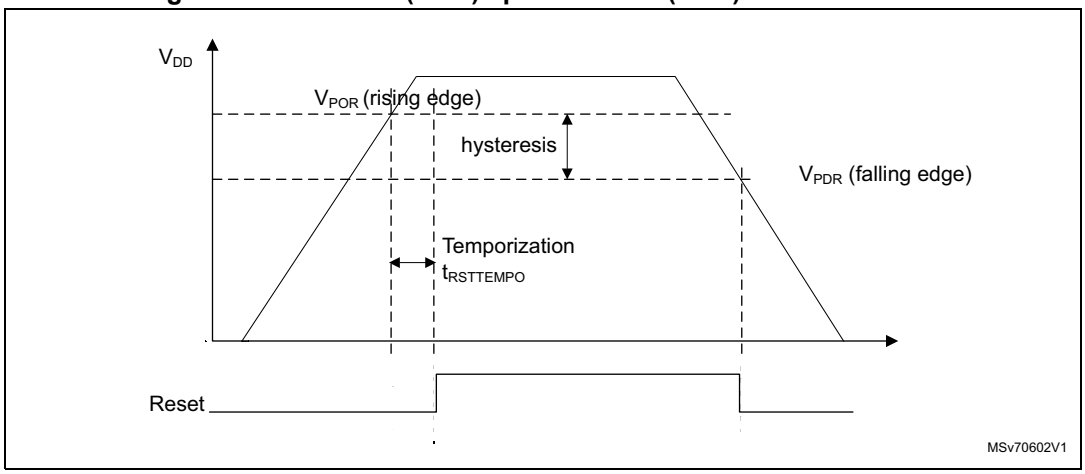
- V_{DD} via POR/PDR (see *Section 10.6.1: Power-on reset (POR)/power-down reset (PDR)*), PVD monitoring (see *Section 10.6.2: Programmable voltage detector (PVD)*), and V_{DD} high threshold monitoring (see *Section 10.6.4: VDD voltage monitoring*)
- V_{DDA} via AVD monitoring (see *Section 10.6.3: Analog voltage detector (AVD)*)
- Temperature monitoring (see *Section 10.6.5: Temperature monitoring*)
- Maximum V_{DD} voltage monitoring (see *Section 10.6.4: VDD voltage monitoring*)

10.6.1 Power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that ensures proper startup operation.

The system remains in reset mode when V_{DD} is below a specified V_{POR} threshold, without the need for an external reset circuit. Once the supply level is above the V_{POR} threshold, the system is taken out of reset (see *Figure 39*). For more details concerning the reset thresholds, refer to the electrical characteristics section of the datasheets.

Figure 39. Power-on (POR) / power-down (PDR) reset waveform



1. For thresholds and hysteresis values, refer to the datasheets.

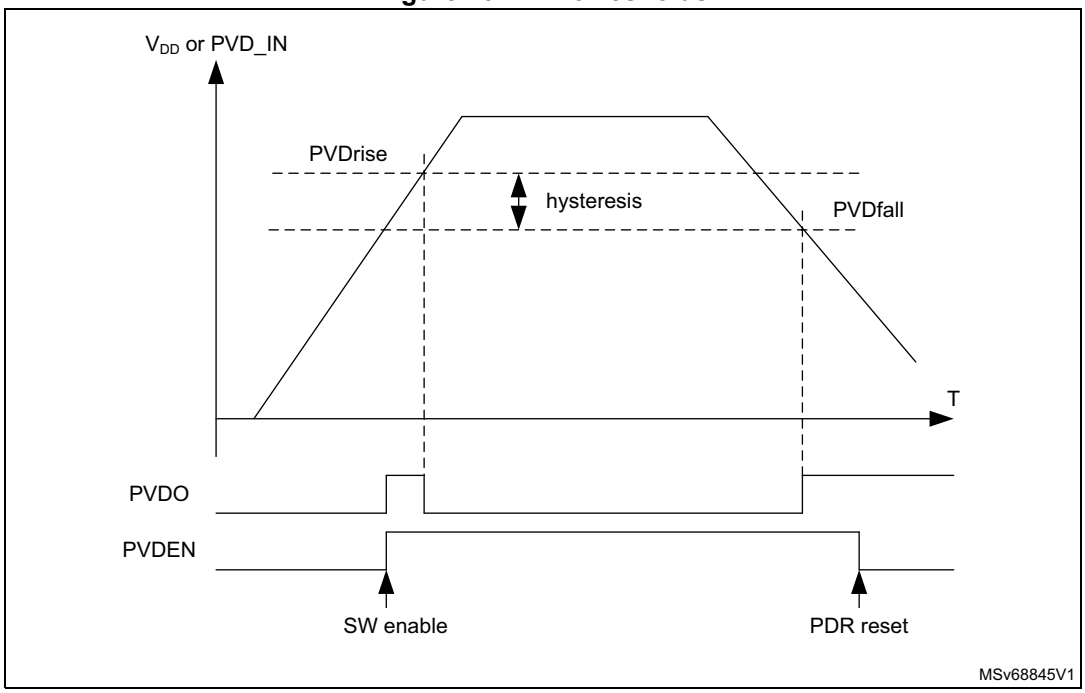
10.6.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[1:0] bits in the *PWR voltage monitor control register (PWR_VMCR)*. The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case, PVD_IN voltage is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDE bit in *PWR voltage monitor control register (PWR_VMCR)*.

A PVDO flag is available in the *PWR voltage monitor status register (PWR_VMSR)* to indicate if the V_{DD} or PVD_IN voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt (PWR_PVD_AVD), provided it has been enabled through the EXTI registers. The rising/falling edge sensitivity of the EXTI line must be configured according to PVD output behavior. As an example, if the EXTI line is configured to rising edge sensitivity, the interrupt is generated when V_{DD} or PVD_IN voltage drops below the PVD threshold. The service routine can then start an emergency shutdown.

Figure 40. PVD thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

10.6.3 Analog voltage detector (AVD)

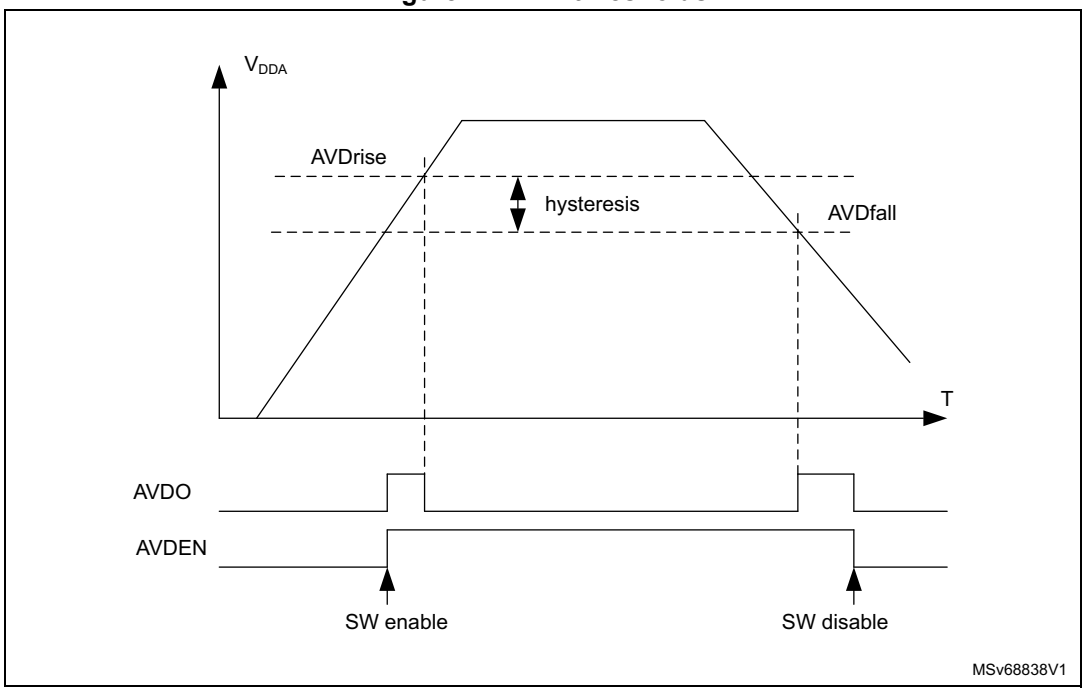
The AVD can be used to monitor the V_{DDA} supply by comparing it to a threshold selected by the ALS bit of the *PWR voltage monitor control register (PWR_VMCR)*.

The AVD is enabled by setting the AVDEN bit in *PWR voltage monitor control register (PWR_VMCR)*.

An AVDO flag is available in the *PWR voltage monitor status register (PWR_VMSR)* to indicate whether V_{DDA} is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers.

The PWR_PVD_AVD interrupt can be generated when V_{DDA} drops below the AVD threshold and/or when V_{DDA} rises above the AVD threshold, depending on EXTI rising/falling edge configuration. As an example, the service routine can indicate when the V_{DDA} supply drops below a minimum level.

Figure 41. AVD thresholds



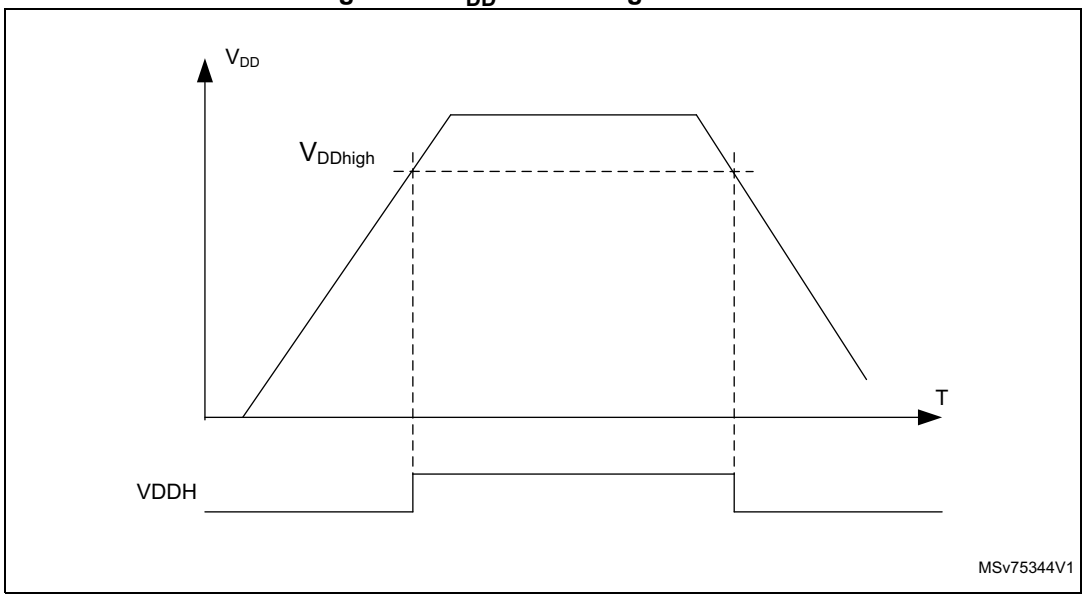
1. For thresholds and hysteresis values, refer to the datasheets.

10.6.4 V_{DD} voltage monitoring

The V_{DD} supply voltage can be monitored by comparing it with the high threshold level V_{DDhigh}. The V_{DD} supply monitoring can be enabled/disabled via MONEN bit in *PWR tamper monitoring control register (PWR_TMCR)*. The VDDH flag of the *PWR tamper monitoring status register (PWR_TMSR)* indicates whether the device V_{DD} voltage is higher or lower than the V_{DDhigh} threshold.

If the V_{DD} voltage monitoring internal tamper is enabled in the TAMP peripheral (ITAMP1E = 1 in the TAMP_CR1 register), a tamper event is generated when the V_{DD} voltage is above the functional range.

Figure 42. V_{DD} monitoring threshold



1. For threshold and hysteresis values, refer to the datasheets.

10.6.5 Temperature monitoring

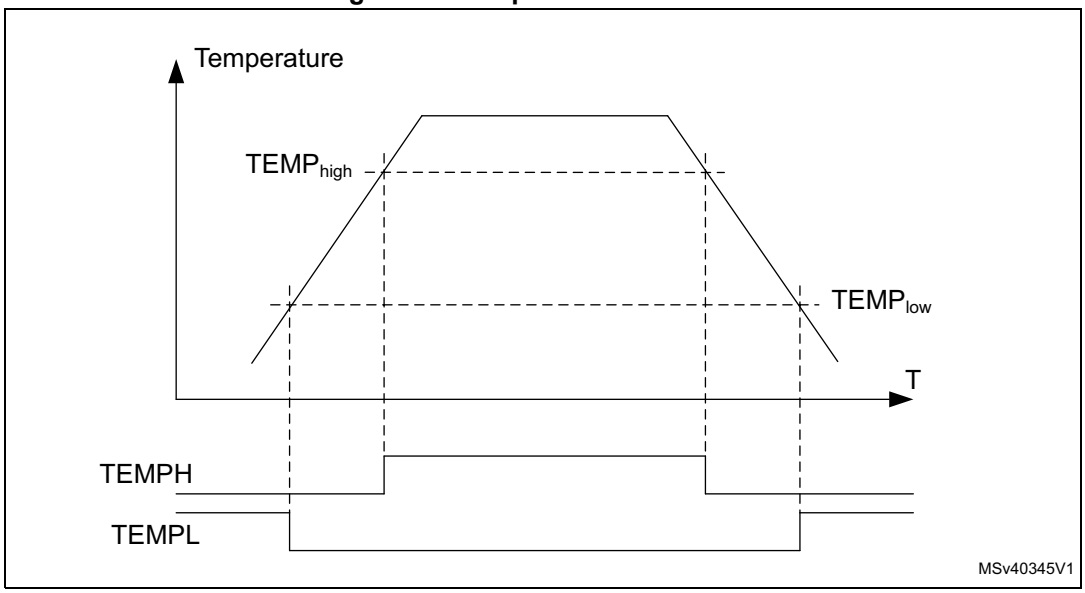
A dedicated temperature sensor cell is embedded in the power control. The junction temperature can be monitored by comparing it with two threshold levels, $TEMP_{high}$ and $TEMP_{low}$. The TEMPH and TEMPL flags of the *PWR tamper monitoring status register (PWR_TMSR)*, which indicate whether the device temperature is higher or lower than the threshold. The temperature monitoring can be enabled/disabled via MONEN bit in *PWR tamper monitoring control register (PWR_TMCR)*.

When enabled, the temperature thresholds increase power consumption. As an example the levels may be used to trigger a routine to perform temperature control tasks.

If the temperature monitoring internal tamper is enabled in the TAMP peripheral (ITAMP2E = 1 in the TAMP_CR1 register), a tamper event is generated when the temperature is above or below the functional range.

TEMPH and TEMPL wake-up interrupts are available on the tamper signals (see *Section 50: Tamper and backup registers (TAMP)*).

Figure 43. Temperature thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

10.7 PWR management

10.7.1 Voltage scaling

The voltage regulator supporting voltage scaling with the following features:

- Run mode voltage scaling
  - VOS0: range 0
  - VOS1: range 1
  - VOS2: range 2
- Stop mode voltage scaling

For more details on voltage scaling values, refer to the product datasheets.

After reset, the system starts on the lowest Run mode voltage scaling (VOS2). The voltage scaling can be changed on-the-fly by software by programming VOS bits of the PWR_VOSCR register, according to the required system performance. When exiting from the Stop mode or Standby mode, the Run mode voltage scaling is reset to the default VOS2 value.

Specific voltage scaling is used in Stop mode to decrease power consumption.

10.7.2 Power management examples

Figure 44. Dynamic voltage scaling in Run mode

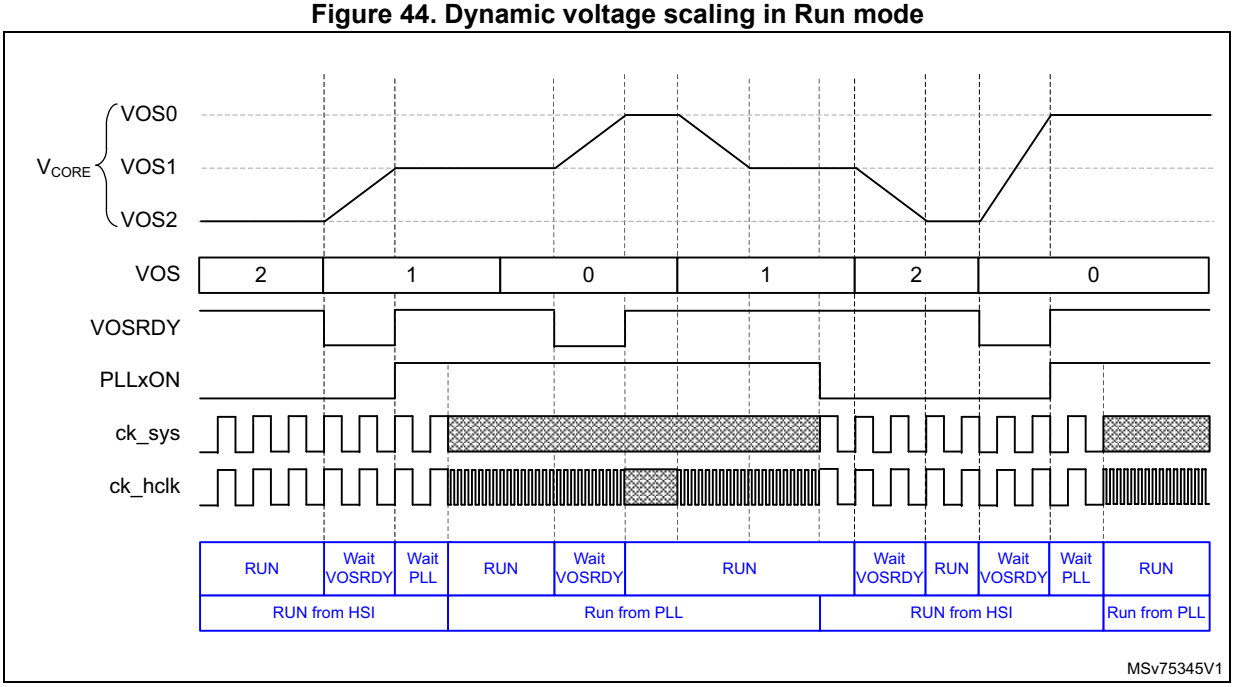


Figure 44 illustrates the following system operation sequence example:

1. After reset, the system starts from HSI with VOS2.
2. The system performance is increased to a medium-speed clock from the PLL with voltage scaling VOS1. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. The system performance is increased to a high-speed clock from the PLL with voltage scaling VOS0. To do this:
   a) Program the voltage scaling to VOS0.
   b) Once the VCORE supply has reached the required level indicated by VOSRDY, increase the clock frequency.
4. The system performance is reduced to a medium-speed clock with voltage scaling VOS1. To do this:
   a) First decrease the system frequency.
   b) Then decrease the voltage scaling to VOS1.
5. The next step is to reduce the system performance to the HSI clock with voltage scaling VOS2. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.
   c) Decrease the voltage scaling to VOS2.

6. The system performance can then be increased to a high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS0.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS must be set accordingly, otherwise the system can be unreliable.

10.8 Power modes

By default, the microcontroller is in Run mode after a system or a power reset. Several low-power modes are available to reduce consumption when there is no need to keep the CPU running, for example when waiting for an external event. The user can select the mode that gives the best compromise between low-power consumption, short startup time, and wake-up sources.

The device features the following low-power modes:

- Sleep mode

  CPU clock off, peripherals such as NVIC and SysTick can run and wake up the CPU when an interrupt or an event occurs. Refer to *Section 10.8.4: Sleep mode*.

- Stop mode

  Achieves the lowest power consumption, while retaining the content of SRAM and registers. All clocks in the core domain are stopped. The PLL, the HSE crystal oscillators, HSI (except if HSIKERON is set), and HSI48 are disabled. The LSE or LSI is still running.

  The RTC can remain active (Stop mode with RTC, Stop mode without RTC).

  The system clock when exiting from Stop mode is set to HSI.

  SRAMs pages can be individually powered-down to further reduce power consumption.

  Refer to *Section 10.8.5: Stop mode*.

- Standby mode

  This mode achieves the lowest power consumption. The internal regulator is switched off so that the core domain is powered off. The PLL, the HSI RC, HSI48, and the HSE crystal oscillators are also switched off.

  The RTC can remain active (Standby mode with RTC, Standby mode without RTC).

  The state of the I/O (except I/Os used by standby mode) during Standby mode can be retained.

  After entering Standby mode, SRAMs and register contents are lost except for registers in the RTC domain and Standby circuitry.

  The device exits Standby mode when an external reset (NRST pin), an IWDG reset, WKUP pin event (configurable rising or falling edge), an RTC event occurs (alarm, periodic wake-up, timestamp), or a tamper detection. The tamper detection can be raised either due to external pins or due to an internal failure detection.

  The system clock after wake-up is HSI at 64 MHz.

  Refer to *Section 10.8.6: Standby mode*.

Table 99 shows the power modes overview.

Table 99. Low-power mode summary

Mode name	Entry	Wake-up source(1)	Wake-up system clock	Effect on clocks	Voltage regulators
Sleep (Sleep-now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt	Same as before entering Sleep mode	– CPU clock OFF – No effect on other clocks or analog clock sources	VOS2, VOS1, or VOS0
	WFE	Wake-up event			
Stop	LPMS = 0 + SLEEPDEEP bit + WFI or Return from ISR or WFE	– Any EXTI line (configured in the EXTI registers)	HSI with the frequency before entering Stop mode, up to 64 MHz	– All clocks OFF except LSI and LSE – HSI can be enabled temporarily when requested by peripherals	Regulator in Stop mode range
Standby	LPMS = 1 + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, IWDG reset, external reset in NRST pin	HSI clock at 64 MHz	All clocks OFF except LSI and LSE	OFF

1. Refer to *Table 100*.

Table 100. Functionalities depending on the working mode(1)

Peripheral	Run	Sleep	Stop		Standby	
			Available	Wake-up capability	Available	Wake-up capability
CPU	Y	-	-	-	-	-
Flash memory	O	O	(2)	-	-	-
SRAM1	Y(3)	Y(3)	O(4)	-	-	-
SRAM2	Y(3)	Y(3)	O(4)	-	-	-
SRAM3	Y(3)	Y(3)	O(4)	-	-	-
XSPI1	O	O	-	-	-	-
Backup registers	Y	Y	Y	-	Y	-
Programmable voltage detector (PVD)	O	O	O	O	-	-
Analog voltage detector (AVD)	O	O	O	O	-	-
GPDMA	O	O	-	-	-	-
High-speed internal (HSI)	O	O	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-
High-speed external (HSE)	O	O	-	-	-	-
Low-speed internal (LSI)	O	O	O	-	O	-

Table 100. Functionalities depending on the working mode(1) (continued)

Peripheral	Run	Sleep	Stop		Standby	
			Available	Wake-up capability	Available	Wake-up capability
Low-speed external (LSE)	O	O	O	-	O	-
Clock security system (CSS)	O	O	-	-	-	-
Clock security system on LSE	O	O	O	O(5)	O	O
V_{DD} voltage and temperature monitoring	O	O	O	O(5)	O	O
RTC/TAMP	O	O	O	O	O	O
Number of TAMP tamper pins	4	4	4	-	3	-
USB, UCPD1	O	O	O	O	-	-
USARTx	O	O	O	O	-	-
Low-power UART (LPUART)	O	O	O	O	-	-
I2Cx	O	O	O	O	-	-
I3C1	O	O	O	O	-	-
Programmable logic array (PLAY1)	O	O	O	O	-	-
SPIx	O	O	O	O	-	-
FDCANx	O	O	-	-	-	-
ADCx	O	O	-	-	-	-
DACx	O	O	O	-	-	-
VREFBUF	O	O	O	-	-	-
Temperature sensor (DTS)	O	O	O	O	-	-
COMPx	O	O	O	O	-	-
OPAMPx	O	O	O	-	-	
Timers (TIMx)	O	O	-	-	-	-
Low-power timer LPTIMx	O	O	O	O	-	-
Independent watchdog (IWDG)	O	O	O	O(5)	O	O
Window watchdog (WWDG)	O	O	-	-	-	-
SysTick timer (SYSTICK)	O	O	O	-	-	-
Position encoder (POSENCx)	O	O	-	-	-	-
high-speed timer (HRTIM1)	O	O	-	-	-	-
CORDIC coprocessor (CORDIC1)	O	O	-	-	-	-
Filter mathematical accelerator (FMACx)	O	O	-	-	-	-
Random number generator (RNG)	O	O	-	-	-	-
AES and secure AES (AES, SAES)	O	O	-	-	-	-
Public key accelerator (PKA)	O	O	-	-	-	-

Table 100. Functionalities depending on the working mode[1] (continued)

Peripheral	Run	Sleep	Stop		Standby	
			Available	Wake-up capability	Available	Wake-up capability
On-the-fly decryption (OTFDEC)	O	O	-	-	-	-
HASH accelerator	O	O	-	-	-	-
CRC calculation unit	O	O	-	-	-	-
GPIOs	O	O	O	O	O[6]	O[7]
EXTI	O	O	O	O	-	-
CCB	O	O	-	-	-	-
RAMCFG	O	O	-	-	-	-
DGB	O	O	-	-	-	-
CACHE	Y	O	-	-	-	-
TZSC	O	O	-	-	-	-
FLASH	Y	Y	-	-	-	-
SBS	O	O	-	-	-	-

1. Y = yes (enabled). O = optional (disabled by default, can be enabled by software). - = not available.
2. The memory can be configured in low-power mode.
3. The SRAM clock can be gated on or off independently. By default, the clock is enabled in Run and Sleep modes.
4. The SRAMs can be powered on or off independently. By default, they are not in power-off mode during Stop.
5. Wake-up with internal tamper.
6. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
7. WKUPx pins are capable of wake-up from Standby mode.

In addition, the power consumption in Run mode can be reduced by slowing down the system clocks, configuring voltage scaling to lower power ranges, and by gating the clocks to the APB and AHB peripherals when they are not used.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is because the Cortex®-M33 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU control registers, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 60.2.5: Debug and low-power modes*.

10.8.1 Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK, HCLK, PCLK) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals before entering the Sleep mode.

For more details, refer to *Section 11: Reset and clock control (RCC)*.

10.8.2 Peripheral clock gating

In Run mode, the HCLK, and PCLK for individual peripherals and memories can be stopped at any time to reduce the power consumption.

In addition, to further reduce the power consumption, the peripheral clocks can be automatically disabled during Sleep mode.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

Disabling the peripheral clocks in Sleep mode is performed automatically by resetting the corresponding bit in the RCC_AHBxLPENR and RCC_APBxLPENR registers.

10.8.3 Low-power modes

Entering low-power modes

The MCU enters in low-power modes by executing the WFI (wait for interrupt), or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M33 system control register is set on return from ISR.

Entering into a low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

Exiting from low-power modes

The MCU exits the Sleep or Stop mode according to how the low-power mode was entered:

- If the WFI instruction or return from ISR was used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the low-power mode, the MCU exits the low-power mode as soon as an event occurs. The wake-up event can be generated either by:
  - An NVIC IRQ interrupt:

    When SEVONPEND = 0 in the Cortex®-M33 system control register, the interrupt must be enabled in the peripheral control register and in the NVIC.

    When the MCU resumes from WFE, the peripheral interrupt pending bit, and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. Only NVIC interrupts with sufficient priority can wake up and interrupt the MCU.

    When SEVONPEND = 1 in the Cortex®-M33 system control register, the interrupt must be enabled in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and, when enabled, the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared.

    All NVIC interrupts wake up the MCU, even the disabled ones.
  - An event:

    An EXTI line must be configured in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit, as the pending bits corresponding to the event line are not set. It may be necessary to clear the interrupt flag in the peripheral.

The MCU exits Standby mode through an external reset (NRST pin), an IWDG reset, a

rising edge on one of the enabled WKUPx pins or a RTC/TAMP event (see *Figure 801: RTC block diagram*).

After waking up from Standby mode, the program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched).

10.8.4 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Entering Sleep mode

The MCU enters the Sleep mode as described in *Entering low-power modes*, when the SLEEPDEEP bit in the Cortex®-M33 system control register is clear (see the table below for details on how to enter the Sleep mode).

Exiting from Sleep mode

The MCU exits the Sleep mode as described in *Exiting from low-power modes* (see the table below for details on how to exit the Sleep mode).

Table 101. Sleep mode

Sleep mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) pending Refer to the Cortex-M33 system control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt pending Refer to the Cortex-M33 system control register.
Mode exit	If WFI or Return from ISR was used for entry Interrupt (see *Table 173: STM32G56x/55x/54x/53x vector table*) If WFE was used for entry and SEVONPEND = 0: Wake-up event (see *Section 19.3: EXTI functional description*) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (see *Table 173: STM32G56x/55x/54x/53x vector table*) or wake-up event (see *Section 19.3: EXTI functional description*)
Wake-up latency	None

10.8.5 Stop mode

The Stop mode is based on the Cortex®-M33 deep-sleep mode combined with the peripheral clock gating. The voltage regulator is configured in Stop mode voltage range. In Stop mode, all clocks in the core domain are stopped. The PLL, HSI, HSI48, and HSE oscillators are disabled.

It is possible to keep the HSI clock enabled during Stop mode, to be quickly available as kernel clock for peripherals.

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduce consumption. The user can select which memory is discarded during Stop mode by means of xxSO bits in PWR_LPMCR registers.

Table 102. Memory power-down block selection for STM32G53x/54x devices

Selection bit	Power-down block in Stop mode
SRAM1PDS	AHB SRAM1 block from 0 to 16 Kbytes
SRAM2PDS1	AHB SRAM2 block from 0 to 16 Kbytes
SRAM2PDS2	AHB SRAM2 block from 16 to 32 Kbytes

Table 103. Memory power-down block selection for STM32G55x/56x devices

Selection bit	Power-down block in Stop mode
SRAM1PD	AHB SRAM1
SRAM2PDS1	AHB SRAM2 block from 0 to 16 Kbytes
SRAM2PDS2	AHB SRAM2 block from 16 to 64 Kbytes
SRAM3PDS	AHB SRAM3

I/O states in Stop mode

In the Stop mode, all I/O pins keep the same state as in the Run mode.

Entering Stop mode

The MCU enters the Stop mode as described in *Entering low-power modes* , when the SLEEPDEEP bit in the Cortex®-M33 system control register is set (see *Table 104* for details on how to enter the Stop mode).

If the flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop mode entry is delayed until the APB access is finished.

In Stop mode, the following features can be selected by programming the individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started, it can be stopped only by a reset (see *Section 47.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in the *RCC RTC domain control register (RCC_RTCCR)*
- The internal RC oscillator LSI clock is configured by the LSION bit in *RCC RTC domain control register (RCC_RTCCR)*.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in *RCC RTC domain control register (RCC_RTCCR)*.

The AVD and the PVD can be used in Stop mode. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx, DACx, OPAMPs, COMPx, the temperature sensor, and the VREFBUF can consume power during the Stop mode, unless they are disabled before entering this mode.

Exiting from Stop mode

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered. Some peripherals are able to wake up the system (refer to *Table 176: EXTI line connections*) from Stop mode.

When exiting Stop mode by issuing an interrupt or a wake-up event, the HSI oscillator is selected as the system clock.

The MCU exits Stop mode by invoking an EXTI interrupt or event depending on how the low-power mode was entered.

When exiting the Stop mode, the MCU is in Run mode, VOS2.

Table 104. Stop mode

Stop mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 000 in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 0 in PWR_LPMCR
	Note: To enter Stop mode, all EXTI line pending bits (in the EXTI rising edge pending register 2 (EXTI_RPR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and the program execution continues.

Table 104. Stop mode (continued)

Stop mode	Description
Mode exit	If WFI or Return from ISR was used for entry: - any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 173: STM32G56x/55x/54x/53x vector table*). If WFE was used for entry and SEVONPEND = 0: - any EXTI line configured in event mode (see *Section 19.3: EXTI functional description*). If WFE was used for entry and SEVONPEND = 1: - any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 173: STM32G56x/55x/54x/53x vector table*). - any EXTI line is configured in event mode (see *Section 19.3: EXTI functional description*)
Wake-up latency	Refer to the datasheet.

10.8.6 Standby mode

The lowest power mode is the Standby mode. It is based on the Cortex®-M33 deep-sleep mode, with the voltage regulators disabled. The PLL, HSI, HSI48, and HSE oscillators are also switched off.

The SRAMs and register contents are lost except for registers in the RTC domain and Standby circuitry (see *Figure 45*).

The POR is always available in Standby mode.

I/O states in Standby mode

In Standby, the I/Os are by default in floating state. If the IORETEN bit in the PWR_IORETR register is set, the I/Os pull or output state is retained during Standby mode. I/O retention mode is enabled for all I/Os except those supporting the standby functionality and JTAG I/Os (PA13, PA14, PA15, and PB4). When entering into Standby mode, the state of the I/O is sampled, and pull-up or pull-down resistor are set to maintain the I/O output during Standby mode.

If the JTAGIORETEN bit in the PWR_IORETR register is set, the I/Os pull or output state is retained during Standby mode. I/O retention mode is enabled for PA13, PA14, PA15, and PB4 (default JTAG pull-up/pull-down after wake-up is not enabled).

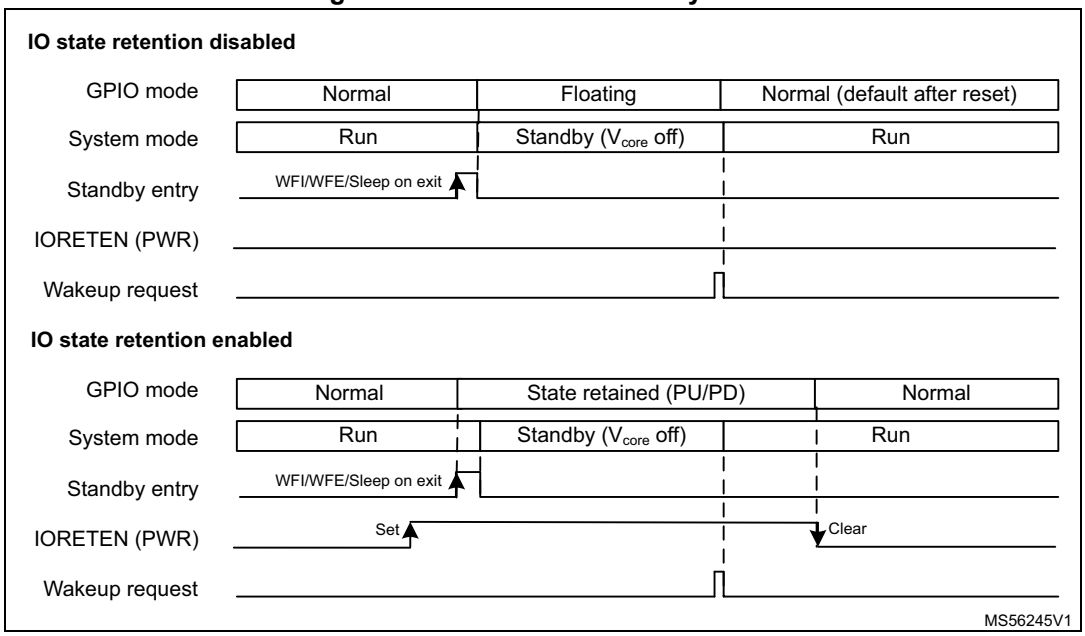
Table 105. I/O retention with pull-up and pull-down

IORETEN/JTAGIORETEN	I/O GPIO configuration before entering Standby	I/O configuration in Standby
0	Any	High-Z

Table 105. I/O retention with pull-up and pull-down (continued)

IORETEN/JTAGIORETEN	I/O GPIO configuration before entering Standby	I/O configuration in Standby
1	Input, no pulls	High-Z
	Input or output with pull-up	Pull-up
	Input or output with pull-down	Pull-down
	Output, no pull driving high level	Pull-up
	Output, no pull driving low level	Pull-down

Figure 45. I/O states in Standby mode



After wake-up from Standby mode, as long as IORETEN (or JTAGIORETEN for JTAG I/Os) is set, the retained stated (pull-up/pull-down) remains applied.

Before Standby, the GPIO pin state can be identified with GPIO_IDR register (when both GPIO port clock and input buffer are enabled).

The application releases the I/O state (clear the retained Pull-up/Pull-down) by clearing the IORETEN (or JTAGIORETEN for JTAG IOs) bit. The GPIOs should be configured before releasing the retained state to control continuously I/O level. It is recommended to clear the IORETEN (or JTAGIORETEN for JTAG IOs) bit just after the required GPIOs are configured (GPIOs which should control continuously I/O level) to avoid pull-up / pull-down conflict set by Standby mode (latched in V_{DD} domain) and set by Run mode.The RTC outputs on PC13 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. Up to seven wake-up (WKUPx, x = 1, 2, 4 to 8) and four tamper pins (two input and two output pins) are available.

Entering Standby mode

The MCU enters the Standby mode as described in *Entering low-power modes*, when the SLEEPDEEP bit in the Cortex®-M33 system control register is set (see *Table 106* for details

on how to enter Standby mode).

In Standby mode, the following features can be selected by programming individual control bits:

- The independent watchdog (IWDG) is started by writing to its Key register or by hardware option. Once started, it can be stopped only by a reset (see *Section 47.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in *RCC RTC domain control register (RCC_RTCCR))*.
- The internal RC oscillator LSI clock is configured by the LSION bit in *RCC RTC domain control register (RCC_RTCCR)*.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in *RCC RTC domain control register (RCC_RTCCR)*.
- The I/Os retention is configured by the IIORETEN/JTAGIORETEN bits in the PWR_IORETR register.

Exiting from Standby mode

The MCU exits the Standby mode as described in *Exiting from low-power modes*. The SBF status flag in the *PWR status register (PWR_PMSR)* indicates that the MCU was in Standby mode. All registers are reset after wake-up from Standby except for *PWR tamper monitoring control register (PWR_TMCR)* and *PWR I/O retention register (PWR_IORETR)* *(see Table 106* for more details on how to exit Standby mode).

When exiting Standby mode, I/Os output state that were retained during Standby through IORETEN bit, keep this configuration upon exiting Standby mode until the IORETEN bit in PWR_IORETR register is cleared. Once IORETEN is cleared, the I/Os are configured to their reset values, or to the pull-up/pull-down state according to the GPIOx_PUPDR registers.

For I/Os, with a pull-up or pull-down predefined after reset (some JTAG/SWD I/Os), in case those pull-up or pull-down are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Also in case the GPIOx_PUPDR values programed after exiting from Standby are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Table 106. Standby mode

Standby mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 1 in PWR_LPMCR – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 1 in PWR_LPMCR – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wake-up source, cleared
Mode exit	WKUPx pin edge, RTC event, external Reset in NRST pin, IWDG Reset, POR reset
Wake-up latency	Reset phase

10.8.7 Power modes output pins

To help the debug, three signals are available as device pins alternate functions:

- **CSLEEP**

  When set, CSLEEP indicates that the system is in Sleep mode: WFI or WFE has been executed.

  When cleared, CSLEEP indicates that the system is in Run mode.

- **CSTOP**

  When set, CSTOP indicates that the system is in CStop mode, meaning that the following conditions are fulfilled:

  – WFI or WFE has been executed with CPU SLEEPDEEP = 1.

  – No AHB/APB clock is running.

  When cleared, CSTOP indicates that the system is not in CStop mode: AHB/APB clocks are running.

The table below explains the MCU power mode depending on these signals states.

Table 107. Power modes output states versus MCU power modes

CSLEEP	CSTOP	MCU power modes(1)
0	0	Run mode
1	0	Sleep mode or Stop mode, with AHB/APB clocks running
1	1	Stop mode

1. CSLEEP and CSTOP are generated in the core domain, consequently they are not driven in Standby mode.

10.9 PWR security and privileged protection

10.9.1 PWR security protection

When the TrustZone® security is activated by the TZEN option byte in the flash memory option byte configuration register, some PWR register fields can be secured against nonsecure accesses.

The PWR TrustZone® security allows the following features to be secured through the PWR_SECCFGR register:

- Low-power mode
- Wake-up (WKUP) pins
- Voltage detection and monitoring
- I/Os retention configuration

Other PWR configuration bits are secure when:

- The system clock selection is secure in RCC: the voltage scaling (VOS) configuration is secure.
- The UCPD is secure in the GTZC: the PWR_UCPDR register is secure.

Table 108 gives a summary of the PWR secured bits following the security configuration bit in PWR_SECCFGR.

A nonsecure access to a secure-protected register bit is denied:

- The secured bits are not written (WI) with a nonsecure write access.
- The secured bits are read as 0 (RAZ) with a nonsecure read access.

A nonsecure write access to PWR_SECCFGR is WI and generates an illegal access event and an interrupt if enabled in the GTZC. It can be read with a nonsecure read access.

When the TrustZone® security is disabled (TZEN = 0xC3), PWR_SECCFGR is RAZ/WI and all other registers are nonsecure.

Table 108. PWR security configuration summary

Secure configuration register	Security configuration bit	Register name	Secured bits	Nonsecure access on secure bits
PWR_SECCFGR	Not applicable(1)	PWR_SECCFGR	All bits	Read OK. WI and illegal access event
PWR_SECCFGR	At least one bit is set	PWR_PRIVCFGR	SPRIV	Read OK. WI
PWR_SECCFGR	LPMSEC	PWR_LPMCR	All bits	WI
PWR_SECCFGR	VUSBSEC	PWR_USBSCR	All bits	WI
PWR_SECCFGR	RTMSEC	PWR_TMCR	All bits	WI
		PWR_RTCCR	All bits	WI
PWR_SECCFGR	RETSEC	PWR_IORETR	All bits	WI

Table 108. PWR security configuration summary (continued)

Secure configuration register	Security configuration bit	Register name	Secured bits	Nonsecure access on secure bits
PWR_SECCFGR	WUPxSEC (x = 1, 2, 4 to 8)	PWR_WUCR	WUPENx	RAZ/WI
			WUPPx	RAZ/WI
			WUPPUPD	RAZ/WI
		PWR_WUSCR	CWUFx	WI
GTZC1_TZSC_SECCFGR2	UCPD1SEC	PWR_UCPDR	All bits	RAZ/WI
RCC_SECCFGR	SYSCLKSEC	PWR_VOSCR	VOS[1:0]	RAZ/WI
PWR_SECCFGR	SCMSEC	PWR_SCCR	All bits	WI
	SCMSEC	PWR_VMCR	All bits	WI

1. PWR_SECCFGR is always secure.

10.9.2 PWR privileged protection

By default, after a reset, all registers can be read or written with both privileged and unprivileged accesses, except PWR_PRIVCFGR, which can be written only with privileged access. PWR_PRIVCFGR can be read by secure and nonsecure, privileged, and unprivileged accesses.

The SPRIV bit in PWR_PRIVCFGR can be written only with secure privileged access. This bit configures the privileged access of all PWR secure functions (defined by PWR_SECCFGR, GTZC, RCC, or GPIO, as shown in *Table 108*).

When the SPRIV bit is set in PWR_PRIVCFGR:

- The PWR secure bits can be written only with privileged access, including PWR_SECCFGR.
- The PWR secure bits can be read only with privileged access except PWR_SECCFGR and PWR_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as 0 and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit of PWR_PRIVCFGR can be written only with privileged access, secure, or nonsecure. It configures the privileged access of all PWR securable functions configured as nonsecure (defined by PWR_SECCFGR, GTZC, RCC, or GPIO, see *Table 108*).

When the NSPRIV bit is set in PWR_PRIVCFGR:

- The PWR securable bits configured as nonsecure can be written only with privileged access.
- The PWR securable bits configured as nonsecure can be read only with privileged access except PWR_PRIVCFGR, which can be read by privileged or unprivileged accesses.
- The VOSRDY and BOOSTRDY bits in PWR_VOSR, PWR_SR, PWR_VMSR, PWR_TMSR and PWR_WUSR can be read with privileged or unprivileged accesses.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as 0, and the write to these bits is ignored (RAZ/WI).

10.10 PWR interrupts

Table 109 gives a summary of the interrupt sources and the way to control them.

Table 109. PWR interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop modes	Exit Standby mode
PWR_PVD_AVD output	Programmable voltage detector through EXTI line 16 (PVD) and EXTI line 53 (AVD)	PVDO/ AVDO	EXTI line 16 enabled (PVD) and EXTI line 53 enabled (AVD)	Set EXTI PIF16 (PVD), or set EXTI PIF53 (AVD)	Yes	No

10.11 PWR registers

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

10.11.1 PWR low power mode control register (PWR_LPMCR)

This register is protected against nonsecure accesses when LPMSEC = 1 in the PWR_SECCFGR register, and against unprivileged accesses when LPMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when LPMSEC = 0 and NSPRIV = 1.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SRAM1 PDS	SRAM2 PDS2	SRAM2 PDS1	SRAM3 PDS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw	rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	Res.	Res.	Res.	LPMS
						rw		rw							rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **SRAM1PDS**: AHB SRAM1 power-down in Stop mode

0: AHB RAM1 content is kept in Stop mode.

1: AHB RAM1 content is lost in Stop mode.

Note: For page definition, refer to c and Table 103: Memory power-down block selection for STM32G55x/56x devices.

Bit 25 **SRAM2PDS2**: AHB SRAM2 page 2 power-down in Stop mode.

0: AHB RAM2 page 2 content is kept in Stop mode.

1: AHB RAM2 page 2 content is lost in Stop mode.

Note: For page definition, refer to Table 102: Memory power-down block selection for STM32G53x/54x devices and Table 103: Memory power-down block selection for STM32G55x/56x devices.

Bit 24 **SRAM2PDS1**: AHB SRAM2 page 1 power-down in Stop mode.

0: AHB RAM2 page 1 content is kept in Stop mode.

1: AHB RAM2 page 1 content is lost in Stop mode.

Note: For page definition, refer to Table 102: Memory power-down block selection for STM32G53x/54x devices and Table 103: Memory power-down block selection for STM32G55x/56x devices.

Bit 23 **SRAM3PDS**: AHB SRAM3 power-down in Stop mode.

0: AHB RAM3 content is kept in Stop mode.

1: AHB RAM3 content is lost in Stop mode.

Note: For page definition, refer to Table 102: Memory power-down block selection for STM32G53x/54x devices and Table 103: Memory power-down block selection for STM32G55x/56x devices.

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FLPS**: flash memory low-power mode in Stop mode

This bit is used to obtain the best trade-off between low-power consumption and restart time when exiting from Stop mode.

When it is set, the flash memory enters low-power mode when the system is in Stop mode.

0: flash memory remains in normal mode when the system enters Stop mode (quick restart time).

1: flash memory enters low-power mode when the system enters Stop mode (low-power consumption).

Bit 8 Reserved, must be kept at reset value.

Bit 7 **CSSF**: clear Standby and Stop flags (always read as 0)

This bit is cleared to 0 by hardware.

0: no effect

1: STOPF and SBF flags cleared

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **LPMS**: low-power mode selection

This bit defines the Deepsleep mode.

0: keeps Stop mode when entering DeepSleep.

1: allows Standby mode when entering DeepSleep.

10.11.2 PWR status register (PWR_PMSR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.
									r	r					

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SBF**: System standby flag

This bit is set by hardware and cleared only by a POR or by setting the CSSF bit.

0: system has not been in Standby mode.

1: system has been in Standby mode.

Bit 5 **STOPF**: Stop flag

This bit is set by hardware and cleared only by any reset or by setting the CSSF bit.

0: system has not been in Stop mode.

1: system has been in Stop mode.

Bits 4:0 Reserved, must be kept at reset value.

10.11.3 PWR voltage scaling control register (PWR_VOSCR)

Some register fields are protected against nonsecure accesses, depending on the RCC_SECCFGR register. These fields can be protected against unprivileged accesses, depending on PWR_PRIVCFGR register configuration.

Address offset: 0x010

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS[1:0]		Res.	Res.	Res.	Res.
										rw	rw				

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **VOS[1:0]**: voltage scaling selection according to performance

These bits control the V_{CORE} voltage level and allow to obtain the best trade-off between power consumption and performance:

- In bypass mode, these bits must be set to VOS[1:0] = 11 (range 0) before increasing the system frequency above 150 MHz.
- When increasing the performance, the voltage scaling must be changed before increasing the system frequency.
- When decreasing performance, the system frequency must first be decreased before changing the voltage scaling.

00: Reserved (no effect)

01: range 2 (VOS2, default): Low-power range

10: range 1: Medium-performance range

11: range 0: High-performance range

Bits 3:0 Reserved, must be kept at reset value.

10.11.4 PWR voltage scaling status register (PWR_VOSSR)

Address offset: 0x014

Reset value: 0x0000 6009

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTVOS[1:0]		ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOSRDY	Res.	Res.	SRAMLS
r	r	r										r			r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **ACTVOS[1:0]**: Current voltage scaling range

This field provides the voltage scaling range value currently applied to V_{CORE}.

00: Reserved

01: VOS2 (lowest power)

10: VOS1

11: VOS0 (highest frequency)

Note: The ACTVOS[1:0] state is not valid in LDO bypass mode (BYPASS = 1). In LDO bypass mode, V_{CORE} is provided from external power supply. The ACTVOS[1:0] state is then always equal to the default VOS2 scale.

Bit 13 **ACTVOSRDY**: Current VOS voltage level ready

0: V_{CORE} is above or below the current voltage scaling provided by ACTVOS[1:0]

1: V_{CORE} is equal to the current voltage scaling provided by ACTVOS[1:0]

Note: The ACTVOSRDY state is not valid in LDO bypass mode (BYPASS = 1). In LDO bypass mode, the ACTVOSRDY state is always set.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **VOSRDY**: Ready bit for V_{CORE} voltage scaling output selection

0: Not ready, voltage level below VOS selected level

1: Ready, voltage level at or above VOS selected level

The VOSRDY flag must be used only when switching from low to high-voltage scale (like switching from VOS2 to VOS0).

Note: The VOSRDY state is not valid in LDO bypass mode (BYPASS = 1). In LDO bypass mode, the VOSRDY state is always set.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SRAMLS**: SRAM memories operation at low speed

0: SRAMs operating at high speed (VOS1 or VOS0 voltage scaling)

1: SRAMs operating at low speed (VOS2 voltage scaling)

10.11.5 PWR tamper monitoring control register (PWR_TMCR)

This register is protected against nonsecure accesses when RTMSEC = 1 in the PWR_SECCFGR register, and against unprivileged accesses when RTMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when RTMSEC = 0 and NSPRIV = 1.

This register is not reset by wake-up from Standby mode, and by a RESET signal. It is reset only by a POR or an RTC domain reset (RTCDRST bit of the RCC_RTCCR register).

Address offset: 0x020

RTC domain reset value: 0x0000 0000

System reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MONEN**: V_{DD} voltage and temperature monitoring enable
0: V_{DD} voltage and temperature monitoring disabled
1: V_{DD} voltage and temperature monitoring enabled

Bit 0 Reserved, must be kept at reset value.

10.11.6 PWR RTC domain control register (PWR_RTCCR)

This register is protected against nonsecure accesses when RTMSEC = 1 in the PWR_SECCFGR register, and against unprivileged accesses when RTMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when RTMSEC = 0 and NSPRIV = 1.

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRTCP
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DRTCP**: Disable RTC domain write protection
In reset state, all the registers in the RTC domain are protected against parasitic write accesses. This bit must be set to enable write access to these registers.
0: Write access to RTC domain disabled
1: Write access to RTC domain enabled

10.11.7 PWR tamper monitoring status register (PWR_TMSR)

This register is not reset by wake-up from Standby mode, or by a RESET signal. It is reset only by a POR or an RTC domain reset (RTCDRST bit of the RCC_RTCCR register).

Address offset: 0x028

RTC domain reset value: 0x0000 0000

System reset value: 0x00X0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	VDDH	Res.	Res.	Res.	Res.	Res.
								r	r	r					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TEMPH**: temperature level monitoring versus high threshold
0: temperature below high threshold level
1: temperature equal or above high threshold level

Bit 22 **TEMPL**: temperature level monitoring versus low threshold
0: temperature above low threshold level
1: temperature equal or below low threshold level

Bit 21 **VDDH**: V_{DD} level monitoring versus high threshold
0: V_{DD} level below high threshold level
1: V_{DD} level equal or above high threshold level

Bits 20:0 Reserved, must be kept at reset value.

10.11.8 PWR USB Type-C® power delivery register (PWR_UCPDR)

This register is protected against nonsecure accesses when UCPD1SEC = 1 in the TZSC_SECCFGR register, and against unprivileged accesses when UCPD1SEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when UCPD1SEC = 0 and NSPRIV = 1.

Note: This register is not available on STM32G53x/54x devices.

Address offset: 0x02C

Reset value: 0x0000 0000

Not affected by exit Standby mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD_STBY	UCPD_DBDIS
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UCPD_STBY**: USB Type-c and Power delivery Standby mode
When set, this bit is used to memorize the UCPD configuration in Standby mode. This bit must be written to 1 just before entering Standby mode when using UCPD, and it must be written to 0 after exiting the standby mode and before writing any UCPD register.

Bit 0 **UCPD_DBDIS**: USB Type-C and power delivery dead battery disable
After exiting reset, the USB Type-C “dead battery” behavior is enabled, which may have a pull-down effect on CC1 and CC2 pins. It is recommended to disable it in all case, either to stop this pull-down or to hand over control to the UCPD (which should therefore be initialized before doing the disable).
0: Enable USB Type-C dead battery pull-down behavior on UCPDx_CC1 and UCPDx_CC2 pins.
1: Disable USB Type-C dead battery pull-down behavior on UCPDx_CC1 and UCPDx_CC2 pins.

10.11.9 PWR supply configuration control register (PWR_SCCR)

This register is protected against nonsecure accesses when SCMSEC = 1 in the PWR_SECCFGR register, and against unprivileged accesses when SCMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when SCMSEC = 0 and NSPRIV = 1.

Address offset: 0x030

Power-on reset value: 0x0000 0X00

System reset value: 0x0000 0X0X

The reset value of this register changes according to the package. Bits 9 and 8 indicate the power configuration. Their values are exclusive.

Reset by POR only, not reset by wake-up from Standby mode and RESET pad. The BYPASS bit of this register is written once after POR. Written-once mechanism locks the register and any further write access is ignored. The system must be power cycled before writing a new value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SMPS EN	LDO EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS
						r	r								rwo

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **SMPSEN**: SMPS enable
The value is set by hardware when the package uses the SMPS regulator.

Bit 8 **LDOEN**: LDO enable
The value is set by hardware when the package uses the LDO regulator.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BYPASS**: power management unit bypass
0: Power management unit normal operation. Use the internal regulator.
1: Power management unit bypassed. Use the external power (voltage monitoring still active)

10.11.10 PWR voltage monitor control register (PWR_VMCR)

This register is protected against nonsecure accesses when SCMSEC = 1 in the PWR_SECCFGR register, and against unprivileged accesses when SCMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when SCMSEC = 0 and NSPRIV = 1.

The PVDE and PLS bits are protected by a lock mechanism. The lock control is in the SBS module controlled b.y PVDL bit in SBS_CFGR2 register. By default, the PVDE, and PLS are unlocked.

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ALS	AVDEN	Res.	Res.	Res.	Res.	Res.	PLS[1:0]		PVDE
						rw	rw						rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **ALS**: analog voltage detector (AVD) level selection
These bits select the analog voltage detector (AVD) thresholds.
0: AVD level0
1: AVD level1

Bit 8 **AVDEN**: peripheral voltage monitor on V_{DDA} enable
0: peripheral voltage monitor on V_{DDA} disabled
1: peripheral voltage monitor on V_{DDA} enabled

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:1 **PLS[1:0]**: programmable voltage detector (PVD) level selection
These bits select the programmable voltage detector (PVD) thresholds.
00: PVD level0
01: PVD level1
10: Reserved
11: PVD_IN pin

Bit 0 **PVDE**: PVD enable
0: PVD disabled
1: PVD enabled

10.11.11 PWR USB supply control register (PWR_USBSCR)

This register is protected against nonsecure accesses when VUSBSEC = 1 in the PWR_SECCFGR register, and against unprivileged accesses when VUSBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when VUSBSEC = 0 and NSPRIV = 1.

Note: This register is not available on STM32G53x/54x devices.

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	USB33SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **USB33SV:** USB supply valid
This bit is used to validate the V_{DD} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the USB peripheral.
0: V_{DD} is not valid for USB usage
1: V_{DD} is valid for USB usage

Bits 24:0 Reserved, must be kept at reset value.

10.11.12 PWR voltage monitor status register (PWR_VMSR)

Address offset: 0x03C

Reset value: 0x00X0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	AVDO	Res.	Res.	Res.
									r			r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PVDO**: programmable voltage detector output

This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.

0: V_{DD} is equal or higher than the PVD threshold selected through the PLS[1:0] bits.

1: V_{DD} is lower than the PVD threshold selected through the PLS[1:0] bits.

Note: Since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **AVDO**: analog voltage detector output on V_{DDA}

This bit is set and cleared by hardware. It is valid only if AVD on VDDA is enabled by the AVDEN bit.

0: V_{DDA} is equal or higher than the AVD threshold selected with the ALS bit.

1: V_{DDA} is lower than the AVD threshold selected with the ALS bit.

Note: Since the AVD is disabled in Standby mode, this bit is equal to 0 after standby or reset until the AVDEN bit is set.

Bits 18:0 Reserved, must be kept at reset value.

10.11.13 PWR wake-up status clear register (PWR_WUSCR)

Each register bit CWUFx (x = 1, 2, 4 to 8) is protected against nonsecure accesses when WUPxSEC = 1 in the PWR_SECCFGR register, and against unprivileged accesses when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC 0 and NSPRIV = 1.

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF 8	CWUF 7	CWUF 6	CWUF 5	CWUF 4	Res.	CWUF 2	CWUF 1
								w	w	w	w	w		w	w

Bits 31:8, 2 Reserved, must be kept at reset value.

Bits 7:3, 1:0 **CWUFx**: clear wake-up pin flag for WUFx (x = 8 to 4, 2 to 1)
These bits are always read as 0.
0: no effect
1: writing 1 clears the WUFx wake-up pin flag (bit is cleared to 0 by hardware).
Note: CWUF8 is not available on the STM32G53x/54x devices.

10.11.14 PWR wake-up status register (PWR_WUSR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF8	WUF7	WUF6	WUF5	WUF4	Res.	WUF2	WUF1
								r	r	r	r	r		r	r

Bits 31:8, 2 Reserved, must be kept at reset value.

Bits 7:3, 1:0 **WUFx**: wake-up pin WUFx flag (x = 8 to 4, 2 to 1)
This bit is set by hardware and cleared only by a RESET pin or by setting the CWUFx bit in PWR_WUSCR register.
0: no wake-up event occurred.
1: wake-up event received from WUFx pin.
Note: WUF8 is not available on the STM32G53x/54x devices.

10.11.15 PWR wake-up configuration register (PWR_WUCR)

Each WUPPUPDx (x = 1, 2, 4 to 8), WUPPx (x = 1, 2, 4 to 8), and WUPENx (x = 1, 2, 4 to 8) bit is protected against nonsecure accesses when WUPxSEC = 1 (x = 1, 2, 4 to 8) in the PWR_SECCFGR register. Each WUPENx bit is protected against unprivileged accesses when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC = 0 and NSPRIV = 1.

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUPPUPD8[1:0]		WUPPUPD7[1:0]		WUPPUPD6[1:0]		WUPPUPD5[1:0]		WUPPUPD4[1:0]		Res.	Res.	WUPPUPD2[1:0]		WUPPUPD1[1:0]	
rw	rw	rw	rw	rw	rw	rw	w	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUP P8	WUP P7	WUP P6	WUP P5	WUP P4	Res.	WUP P2	WUP P1	WUP EN8	WUP EN7	WUP EN6	WUP EN5	WUP EN4	Res.	WUPE N2	WUP EN1
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 21:20, 10, 2 Reserved, must be kept at reset value.

Bits 31:22, 19:16 **WUPPUPDx[1:0]**: wake-up pin pull configuration for WKUPx (x = 8 to 4, 2 to 1)

These bits define the I/O pad pull configuration used when WUPENx = 1. The associated GPIO port pull configuration must be set to the same value or to 00. The wake-up pin pull configuration is kept in Standby mode.

00: no pull-up

01: pull-up

10: pull-down

11: reserved

Note: WUPPUPD8 is not available on the STM32G53x/54x devices.

Bits 15:11, 9:8 **WUPPx**: wake-up pin polarity bit for WUPx (x = 8 to 4, 2 to 1)

These bits define the polarity used for event detection on WUPx external wake-up pin.

0: detection on high level (rising edge)

1: detection on low level (falling edge)

Note: WUPP8 is not available on the STM32G53x/54x devices.

Bits 7:3, 1:0 **WUPENx**: enable wake-up pin WUPx (x = 8 to 4, 2 to 1)

These bits are set and cleared by software.

0: an event on WUPx pin does not wake-up the system from Standby mode.

1: a rising or falling edge on WUPx pin wakes up the system from Standby mode.

Note: An additional wake-up event is detected if WUPx pin is enabled (by setting the WUPENx bit) when WUPx pin level is already high when WUPPx selects rising edge, or low when WUPPx selects falling edge.

WUPEN8 is not available on the STM32G53x/54x devices.

10.11.16 PWR I/O retention register (PWR_IORETR)

This register is protected against nonsecure accesses when RETSEC = 1 in the PWR_SECCFGR register, and against unprivileged accesses when RETBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when RETSEC = 0 and NSPRIV = 1.

Address offset: 0x050

Power-on reset value: 0x0000 0000

System reset value: 0x000X 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JTAGIORETEN
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IORETEN
															rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **JTAGIORETEN**: JTAG IOs retention enable

0: PA13, PA14, PA15, and PB4 IO Retention mode disabled.

1: PA13, PA14, PA15, and PB4 Retention mode enabled.

The I/O pull or output level is sampled when entering Standby mode. Pull-up or pull-down are applied to the IO during the Standby mode, according to the I/O pull or sampled level.

Note: The IO state is not retained if the DBG_STANDBY bit is set in DBGMCU_CR register.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **IORETEN**: IO retention enable

0: IOs (except JTAG I/Os) Retention mode disabled.

1: IOs (except JTAG I/Os) Retention mode enabled for all IOs except those that are functional in Standby mode.

The I/O pull or output level is sampled when entering Standby mode. Pull-up or pull-down are applied to the IO during the Standby mode, according to the I/O pull or sampled level.

Note: The IO state is not retained if the DBG_STANDBY bit is set in DBGMCU_CR register.

10.11.17 PWR security configuration register (PWR_SECCFGR)

This register can be written only when the accesses is secure. It can be read by secure or nonsecure accesses. The register is write-protected against unprivileged write accesses when SPRIV = 1 in the PWR_PRIVCFGR.

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VUSB SEC	RTM SEC	SCM SEC	LPM SEC	RET SEC	Res.	Res.	Res.	WUP8 SEC	WUP7 SEC	WUP6 SEC	WUP5 SEC	WUP4 SEC	Res.	WUP2 SEC	WUP1 SEC
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw		rw	rw

Bits 31:16, 10:8, 2 Reserved, must be kept at reset value.

Bit 15 **VUSBSEC**: voltage USB secure protection

0: PWR_USBSCR can be read and written with secure or nonsecure accesses.

1: PWR_USBSCR can be read and written only with secure accesses.

Bit 14 **RTMSEC**: RTC domain and tamper monitoring secure protection

0: PWR_TMCR and PWR_RTCCR can be read and written with secure or nonsecure accesses.

1: PWR_TMCR and PWR_RTCCR can be read and written only with secure accesses.

Bit 13 **SCMSEC**: supply configuration and monitoring secure protection.

0: PWR_SCCR and PWR_VMCR can be read and written with secure or nonsecure accesses.

1: PWR_SCCR and PWR_VMCR can be read and written only with secure accesses.

Bit 12 **LPMSEC**: low-power modes secure protection

0: PWR_LPMCR can be read and written with secure or nonsecure accesses.

1: PWR_LPMCR can be read and written only with secure accesses.

Bit 11 **RETSEC**: retention secure protection

0: PWR_IORETR can be read and written with secure or nonsecure accesses.

1: PWR_IORETR can be read and written only with secure accesses.

Bits 10:8 Reserved, must be kept at reset value.

Bits 7:3, 1:0 **WUPxSEC**: WUPx secure protection (x = 8 to 4, 2 to 1)

0: The bits related to the WKUPx wake-up pin in PWR_WUSCR and PWR_WUCR can be read and written with secure or nonsecure accesses.

1: The bits related to the WKUPx wake-up pin in PWR_WUSCR and PWR_WUCR can be read and written only with secure accesses.

Note: WUP8SEC is not available on the STM32G53x/54x devices.

10.11.18 PWR privilege configuration register (PWR_PRIVCFGR)

This register can be written only when the access is privileged. It can be read by privileged or unprivileged accesses.

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: PWR nonsecure functions privilege configuration

Set and reset by software. This bit can be written only by privileged accesses, secure or nonsecure.

0: Read and write to PWR nonsecure functions can be done by privileged or unprivileged accesses.

1: Read and write to PWR nonsecure functions can be done by privileged accesses only.

Bit 0 **SPRIV**: PWR secure functions privilege configuration

Set and reset by software. This bit can be written only by a secure privileged accesses.

0: Read and write to PWR secure functions can be done by privileged or unprivileged accesses.

1: Read and write to PWR secure functions can be done by privileged accesses only.

10.11.19 PWR register map

Table 110. PWR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PWR_LPMCR	Res.	Res.	Res.	Res.	Res.	SRAM1PDS	SRAM2PDS2	SRAM2PDS1	SRAM3PDS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	Res.	Res.	Res.	LPMS
	Reset value						0	0	0	0														0		0							0
0x04	PWR_PMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.
	Reset value																										0	0					
0x008 to 0x00F	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x10	PWR_VOSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS [1:0]		Res.	Res.	Res.	Res.
	Reset value																											0	1				
0x14	PWR_VOSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACTVOS [1:0]		ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOSRDY	Res.	Res.	SRAMLS
	Reset value																	0	1	1										1			1
0x018 to 0x01F	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x20	PWR_TMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	Res.
	Reset value																															0	
0x24	PWR_RTCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRTCP
	Reset value																																0
0x28	PWR_TMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	VDDH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0																					
0x2C	PWR_UCPDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD_STBY	UCPD_DBDIS
	Reset value																															0	0
0x30	PWR_SCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPSEN	LDOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS
	Reset value																							X	X								0
0x34	PWR_VMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALS	AVDEN	Res.	Res.	Res.	Res.	Res.	PLS[1:0]		PVDE
	Reset value																							0	0							0	0
0x38	PWR_USBSCR	Res.	Res.	Res.	Res.	Res.	Res.	USB33SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0																									

Table 110. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3C	PWR_VMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDD	Res.	Res.	AVDD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0			0																			
0x40	PWR_WUSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF8(1)	CWUF7	CWUF6	CWUF5	CWUF4	Res.	CWUF2	CWUF1
	Reset value																									0	0	0	0	0		0	0
0x44	PWR_WUSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF8(1)	WUF7	WUF6	WUF5	WUF4	Res.	WUF2	WUF1
	Reset value																									0	0	0	0	0		0	0
0x48	PWR_WUCR	WUPPUPD8 [1:0](1)		WUPPUPD7 [1:0]		WUPPUPD6 [1:0]		WUPPUPD5 [1:0]		WUPPUPD4 [1:0]		Res.	Res.	WUPPUPD2 [1:0]		WUPPUPD1 [1:0]		WUPP8(1)	WUPP7	WUPP6	WUPP5	WUPP4	Res.	WUPP2	WUPP1	WUPEN8(1)	WUPEN7	WUPEN6	WUPEN5	WUPEN4	Res.	WUPEN2	WUPEN1
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0
0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x50	PWR_IORETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JTAGIORETEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IORETEN
	Reset value																0																0
0x054 to 0x0FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x100	PWR_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		VUSBSEC	RTMSEC	SCMSEC	LPMSEC	RETSEC	Res.	Res.	Res.	WUP8SEC(1)	WUP7SEC	WUP6SEC	WUP5SEC	WUP4SEC	Res.	WUP2SEC	WUP1SEC
	Reset value																	0	0	0	0	0				0	0	0	0	0		0	0
0x104	PWR_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
	Reset value																															0	0

1. This bit is not available on the STM32G53x/54x devices.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

11 Reset and clock control (RCC)

This section describes the RCC implementation on STM32G53x/54x and STM32G55x/56x devices. Refer to the datasheet for implementation details.

11.1 RCC introduction

The RCC block manages the clock and reset generation for the whole microcontroller.

The operating modes to which this section refers are defined in *Section 10.8: Power modes*.

11.2 RCC main features

- AHB-lite bus interface
- Reset block:
  - Generation of local and system reset
  - Bidirectional pin reset allowing the reset of the microcontroller and external devices
  - WWDG and IWDG reset supported
  - Power-on (POR) resets initiated by the power control (PWR)
- Clock generation block:
  - Generation and dispatching of clocks for the whole microcontroller
  - Up to two PLLs[a] using integer or fractional ratios:
    Update of PLL fractional ratios on-the-fly
    Each PLL outputs three clocks
  - Smart clock gating to reduce power consumption
  - Two external oscillators:
    High-speed external oscillator (HSE) supporting a wide range of crystals from 4 to 50 MHz frequency
    Low-speed external oscillator (LSE) for the 32.768 kHz crystals
  - Several internal oscillators:
    HSI oscillator delivering up to 64 MHz clock
    HSI48 oscillator delivering a 48 MHz clock[b]
    LSI oscillator delivering a 32 kHz very-low power clock
  - Buffered clock outputs (MCO1/2)
  - Generation of three interrupts lines:
    Two general interrupt lines for ready and unlock events
    One dedicated interrupt lines for LSE and HSE clock security events

a. PLL2 is not available on STM32G53x/54x devices. Refer to the device datasheet for the number of PLL instances.

b. HSI48 is not available on STM32G53x/54x devices. Refer to the device datasheet for the number of PLL instances.

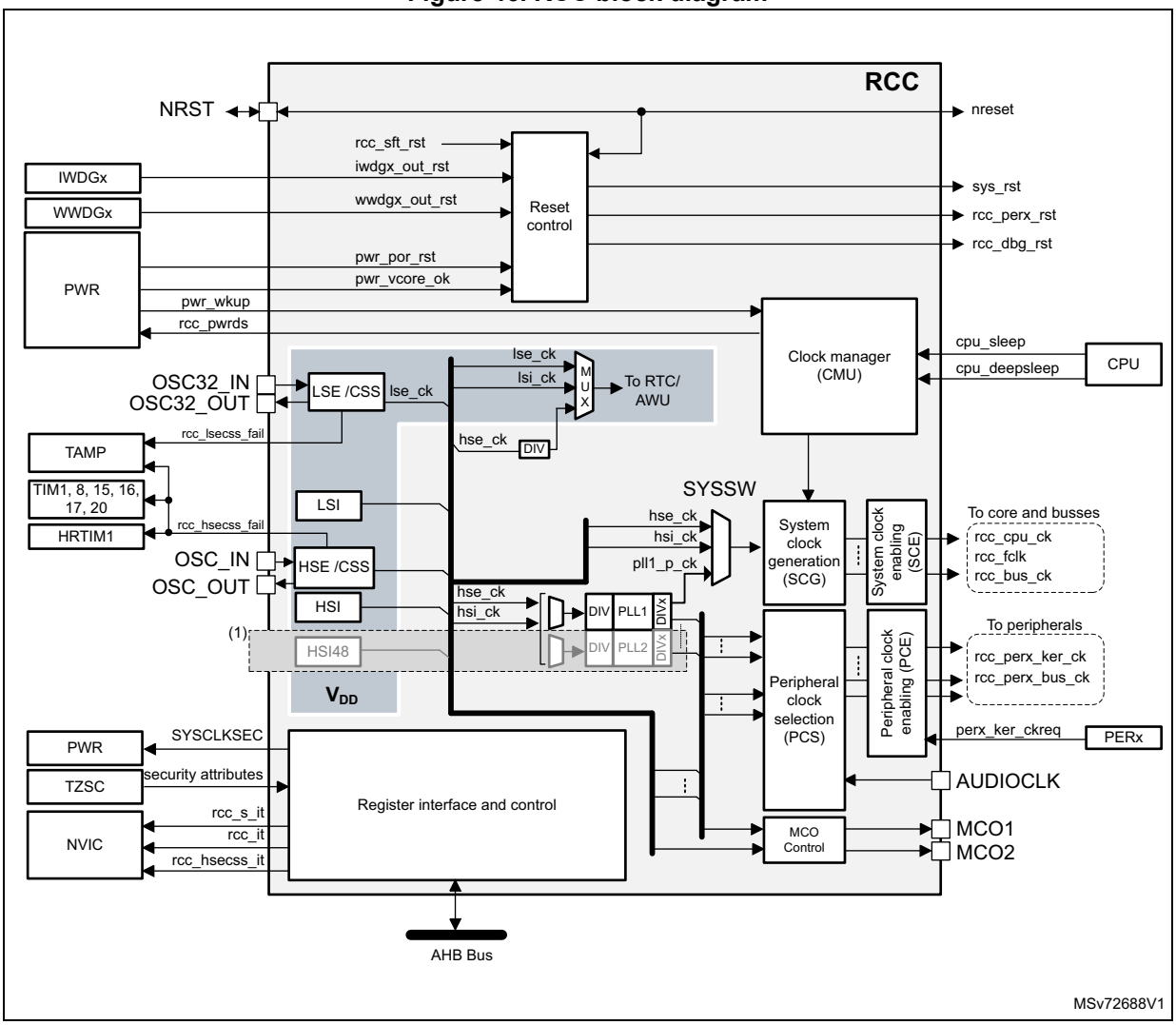
  - Clock generation handling in Stop and Standby modes
- TrustZone®:
  - Restricted access to RCC resources according to security and privilege attributes

11.3 RCC functional description overview

11.3.1 RCC block diagram

shows the RCC block diagram.

Figure 46. RCC block diagram



1. PLL2 and HSI48 are not available on STM32G53x/54x devices. Refer to the device datasheet for more details.

11.3.2 RCC pins and internal signals

lists the RCC inputs and output signals connected to package pins or balls.

Table 111. RCC input/output signals connected to package pins or balls

Signal name	Signal type	Description
NRST	I/O	System reset. It can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input
OSC_OUT	O	System oscillator output
MCO1	O	Output clock for external devices
MCO2	O	Output clock for external devices
AUDIOCLK	I	External kernel clock input for SPI blocks

Table 112. RCC internal input/output signals

New signal name	Signal type	Description
rcc_it	O	General interrupt request line
rcc_s_it	O	General secure interrupt request line
rcc_hsecss_it	O	HSE clock security failure interrupt
rcc_hsecss_fail	O	Event indicating that a HSE clock security failure is detected
rcc_lsecss_fail	O	Event indicating that a LSE clock security failure is detected
nreset	I/O	Application reset
sys_rst	I/O	System reset
iwdg_out_rst	I	Reset line driven by the IWDG, indicating that a timeout occurred
wwdg_out_rst	I	Reset line driven by the WWDG, indicating that a timeout occurred
pwr_por_rst	I	Power-on reset generated by the PWR block
rcc_perx_rst	O	Reset generated by the RCC for the peripherals
pwr_wkup	I	Wakeup domain request generated by the PWR and used to restore the domain clocks
rcc_pwd_req	O	Low-power request generated by the RCC and used to set the device into low-power mode
cpu_sleep	I	Signals generated by the CPU, indicating if the CPU is in CRun, Sleep or CStop
cpu_deepsleep	I	
perx_ker_ckreq	I	Signal generated by some peripherals in order to request the activation of their kernel clock

Table 112. RCC internal input/output signals (continued)

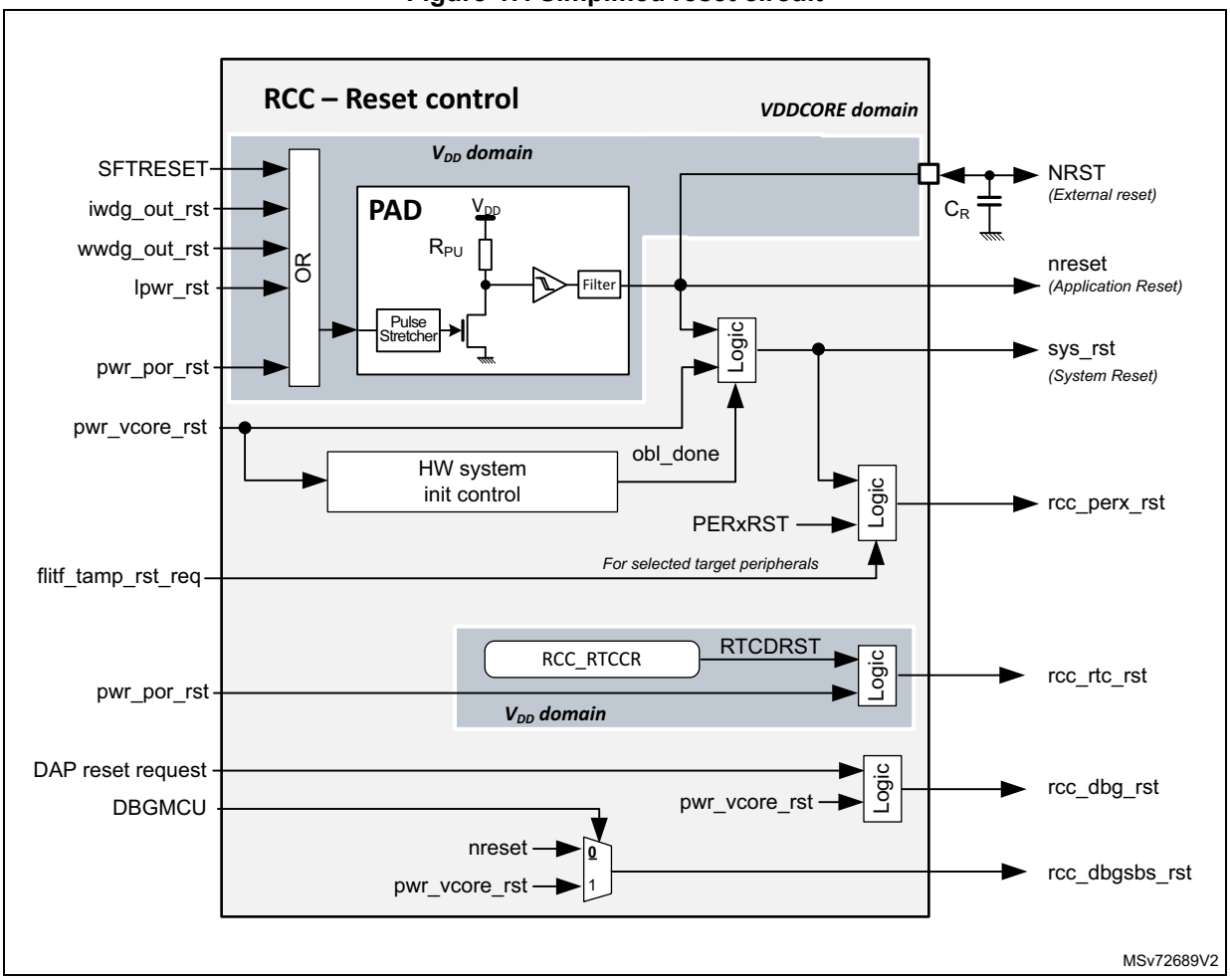
New signal name	Signal type	Description
rcc_perx_ker_ck	O	Kernel clock signals generated by the RCC, for some peripherals
rcc_perx_bus_ck	O	Bus interface clock signals generated by the RCC for peripherals
rcc_bus_ck	O	Clocks for APB, AHB and AXI bridges generated by the RCC
rcc_cpu_ck	O	Clocks for the CPU, generated by the RCC
rcc_fclk	O	

11.4 RCC reset functional description

The RCC handles several types of resets:

- Reset from/to external devices (NRST)
- Application reset (nreset)
- System reset (sys_rst)
- RTC domain reset (rcc_rtc_rst)
- Debug domain reset (rcc_dbg_rst)
- Dedicated peripheral resets (rcc_perx_rst)

Figure 47. Simplified reset circuit



11.4.1 Reset from/to external devices (NRST pin)

The RCC offers a bidirectional NRST pad, which can be asserted either by internal reset sources or by external components.

A pulse stretcher guarantees a minimum reset pulse duration of 20 µs (refer to the device datasheet for details). In addition, the NRST assertion can be extended thanks to the C_R capacitor. A filter is also present to suppress spurs coming from the NRST pad.

The NRST is active low and can be activated by one of the following sources:

- An external component connected to NRST
- The **pwr_por_rst** signal from the PWR block

  The power-on/off reset signal (pwr_por_rst) is asserted by the PWR when the V_{DD} supply is lower than the V_{POR} threshold. The V_{POR} threshold is defined to reset the product when the supply voltage is no longer safe. Refer to *Section 10: Power control (PWR)* for details.
- An independent watchdog reset (**iwdg_out_rst**)
- A window watchdog reset (**wwdg_out_rst**)
- A software reset from the CPU core

  The CPU can generate a system reset via the SYSRESETREQ bit (also named SFTRESET) issued by the Cortex®-M33 core. The SYSRESETREQ bit of the Cortex®-M33 core is controlled via the AIRCR register. Refer to the Arm® technical reference manual for more details (see http://infocenter.arm.com).
- A reset from the low-power mode security reset, depending on option byte configuration (**lpwr_rst**). Refer to *Section 11.4.7: Low-power mode security reset (lpwr_rst)*.

Note: When the NRST pad is not used, it is recommended to connect this pin to the ground via a 10 to 100 nF capacitor (C_R in Figure 46).

Refer to *Table 113: Reset coverage summary* for details.

11.4.2 Application and system resets

The application reset (**nreset**) is activated only when the NRST pad is asserted.

A system reset (**sys_rst**) resets most of the registers to their default values unless otherwise specified in the register description.

A system reset is activated when:

- NRST pad is asserted (that is, **nreset**),
- Exiting from Standby (**pwr_vcore_rst**)

  The core reset signal (**pwr_vcore_rst**) is asserted when the V_{DDCORE} or V_{DD} supply is not valid. As V_{DDCORE} is switched OFF when the product is in Standby, **pwr_vcore_rst** is also asserted in Standby. The registers outside the V_{DDCORE} domain (RTC, WKUP, IWDG, and GPIO pull-up/pull-down configuration during Standby and Standby mode exit) are not impacted.

11.4.3 RTC domain reset

The RTC domain contains a group of peripherals powered by V_{DD}. The RCC offers the possibility to perform a dedicated reset to the peripherals located in this domain. This domain contains the RTC, the TAMP, as well as the LSE and its logic.

There are two ways to generate an RTC domain reset:

- By a software reset

  The software reset is triggered by setting the RTCDRST bit of *RCC RTC domain control register (RCC_RTCCR)*. Write access to this bit must be enabled via the DRTCP bit located into the PWR block.
- By asserting the **pwr_por_rst** reset signal. This signal is asserted when the V_{DD} supply is lower than the minimum allowed V_{DD} value.

11.4.4 CoreSight™ debug reset

There are two ways to reset the CoreSight™ debug components:

- Over the DAP by setting the bit CDBGRSTREQ of the control/status register of the debug port.

  This action asserts the debug reset request signal cdbgrstreq connected to the RCC. Then the RCC activates the debug reset (**rcc_dbg_rst**), a handshake signal cdbgrstack is provided to the DAP request. The debug reset remains as long as the cdbgrstreq is set (refer to *Figure 47* for details).
- When a V_{DDCORE} power-on reset occurs (**rcc_vcore_rst**)

  This reset is activated after a power-on reset, or when the product exits from Standby.

11.4.5 Option bytes loading

As shown in *Figure 47*, the option byte loading (OBL) sequence is triggered under the following conditions:

- After a power-on, or
- When the system exits from Standby

The system reset (**sys_rst**) can be released only when the option byte loading (OBL) is complete.

11.4.6 Peripherals reset

The application can reset individually most of the peripherals whenever it is requested. Most of the time, this is done via the PERxRST bitfields.

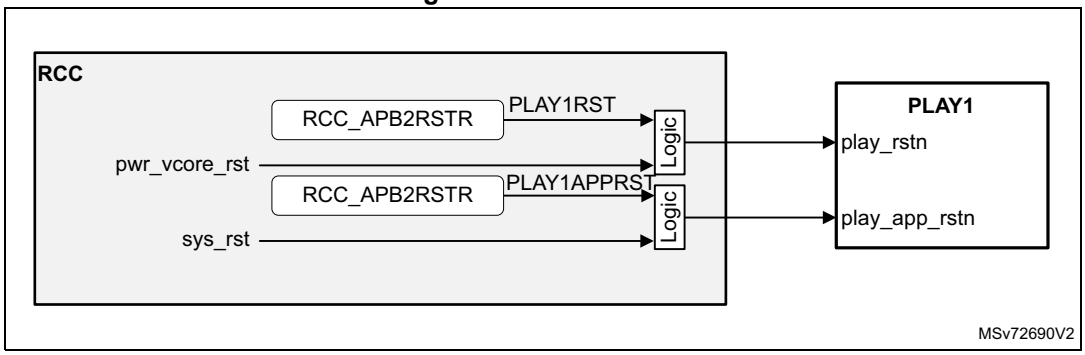
To reset a peripheral, the corresponding reset bit must be set, and then clears. There is no need to enable the peripheral clocks for resetting a peripheral.

Also note that the CRYP, SAES, and HASH blocks can be reset in case of a tamper event.

Specific case of PLAY1 reset[a]

The PLAY1APPRST bit resets the PLAY1 but not the PLAY1 configuration. When the reset is released, the PLAY1 restarts. The PLAY1RST bit resets the complete PLAY1 block including its configuration.

Figure 48. PLAY1 reset



11.4.7 Low-power mode security reset (lpwr_rst)

To avoid that critical applications mistakenly enter a low-power mode, the following low-power mode security resets are available. If enabled in option bytes, an NRST reset is generated when any of the following conditions occurs:

- Standby mode entry: this type of reset is enabled by resetting the nRST_STDBY bit in user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
- Stop mode entry: this type of reset is enabled by resetting the nRST_STOP bit in user option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.

For further information on the user option bytes, refer to *Section 7.4.1: About option bytes*.

a. PLAY1 is not available on the STM32G53x/54x devices. Refer to the device datasheet for details.

11.4.8 Reset coverage summary

Table 113 gives a detailed view of the reset coverage for the most important reset sources.

Note: When V_{DD} is not valid, V_{DDCORE} is not valid as well.

Table 113. Reset coverage summary (1)

Reseted functions		Main reset lines						
		pwr_por_rst (2)	pwr_vcore_rst (3)	sys_rst	nreset (4)	rcc_dbg_rst	rcc_perx_rst	rcc_rtc_rst
CPU		-	-	x	-	-	-	-
WWDG		-	-	x	-	-	-	-
IWDG		-	-	-	x	-	-	-
AHB/APB Interconnections		-	-	x	-	-	-	-
Debug components, including DBGMCU Reset all the debug parts except the SWJ-DP function. The SWJ-DP is reset by the NJTRST or **pwr_vcore_rst** resets.		-	-	-	-	x	-	-
Hardware system initialization. It includes the reload of the option bytes.		x	x	-	-	-	-	-
RCC	HSE, HSI related registers	-	-	x	-	-	-	-
	LSI oscillator and related registers	x	-	-	x	-	-	-
	RCC_RTCCR, RTCSEL of RCC_RTCCFGR, RTCEN of RCC_RTCCFGR	-	-	-	-	-	-	x
	Other RCC registers	-	-	x	-	-	-	-
PWR	PWR_TMCR, PWR_TMSR	x	-	-	-	-	-	x
	PWR_SCCR	x	-	-	-	-	-	-
	PWR_WUSR	x	-	-	x	-	-	-
	Other registers	-	-	x	-	-	-	-
GPIO	Pull-up/pull-down configuration during Standby and Standby mode exit	-	-	x	-	-	x	-
LSE		x	-	-	-	-	-	x
RTC	Including backup registers	x	-	-	-	-	-	x
PLAY1 (5)		-	x	x	-	-	x	-
Kernel clock switches		-	-	x	-	-	-	-
Other Peripherals		-	-	x	-	-	x	-

1. The ‘x’ means that the function is reset by the corresponding reset line, the ‘-’ means that the function is not reset by the corresponding reset line.
2. The **pwr_por_rst** is asserted when the voltage applied to V_{DD} is not valid.
When **pwr_por_rst** is asserted, the **pwr_vcore_rst**, the **rcc_rtc_rst**, NRST, the **sys_rst**, and the **nreset**, are asserted as well.
3. The **pwr_vcore_rst** is asserted when the voltage applied to V_{DD} is not valid, or when the system exits from Standby (because VDDCORE was switched OFF).
When **pwr_vcore_rst** is asserted, the **sys_rst**, and the **pwr_dbg_rst** are asserted as well.
4. When **nreset** is asserted, the **sys_rst** is asserted as well.
5. PLAY1 is not available on the STM32G53x/54x devices.

11.4.9 Reset source identification

The application can identify the reset source by checking the reset flags of the *RCC reset status register (RCC_RSR)*.

The software can reset the flags by setting the RMVF bit.

Table 114 shows how the status bits of the *RCC reset status register (RCC_RSR)* behave according to the situation that generated the reset. For example, when an IWDG timeout occurs, if the CPU is reading the *RCC reset status register (RCC_RSR)* during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

Table 114. Reset source identification (RCC_RSR)(1)

	Reset	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF
1	Power-on reset	0	0	0	0	1	1
2	Pin/pad reset	0	0	0	0	0	1
3	System reset generated by CPU	0	0	0	1	0	1
4	WWDG reset	0	1	0	0	0	1
5	IWDG reset	0	0	1	0	0	1
6	Illegal stop entry reset	1	0	0	0	0	1

1. Gray cells highlight the register bits that are set.

11.5 RCC clock functional description

The RCC provides a wide choice of clock generators:

- HSI (high-speed internal oscillator) clock: ~ 8, 16, 32 or 64 MHz
- HSE (high-speed external oscillator) clock: 4 to 50 MHz
- LSE (low-speed external oscillator) clock: 32.768 kHz
- LSI (low-speed internal oscillator) clock: ~ 32 kHz
- HSI48 (high-speed internal oscillator) clock: ~48 MHz(a)

The RCC offers a high flexibility for the application to select the appropriate clock for CPU and peripherals.

To optimize the power consumption, each clock source can be switched ON and OFF independently.

The RCC provides up to two PLLs, each of them can be configured in integer or fractional mode.[a]

As shown below, the RCC offers two clock outputs (MCO1 and MCO2), with a great flexibility in the clock selection and frequency adjustment.

The SCG block (system clock generation) includes several prescalers used to configure the CPU and bus matrix clock frequencies.

The PCS block (peripheral clock selection) provides several dynamic switches allowing a large choice of kernel clock distribution to peripherals.

The PCE and SCE blocks (peripheral or system clock enable) perform the peripheral kernel clock gating, and the bus interface/cores/bus matrix clock gating, respectively.

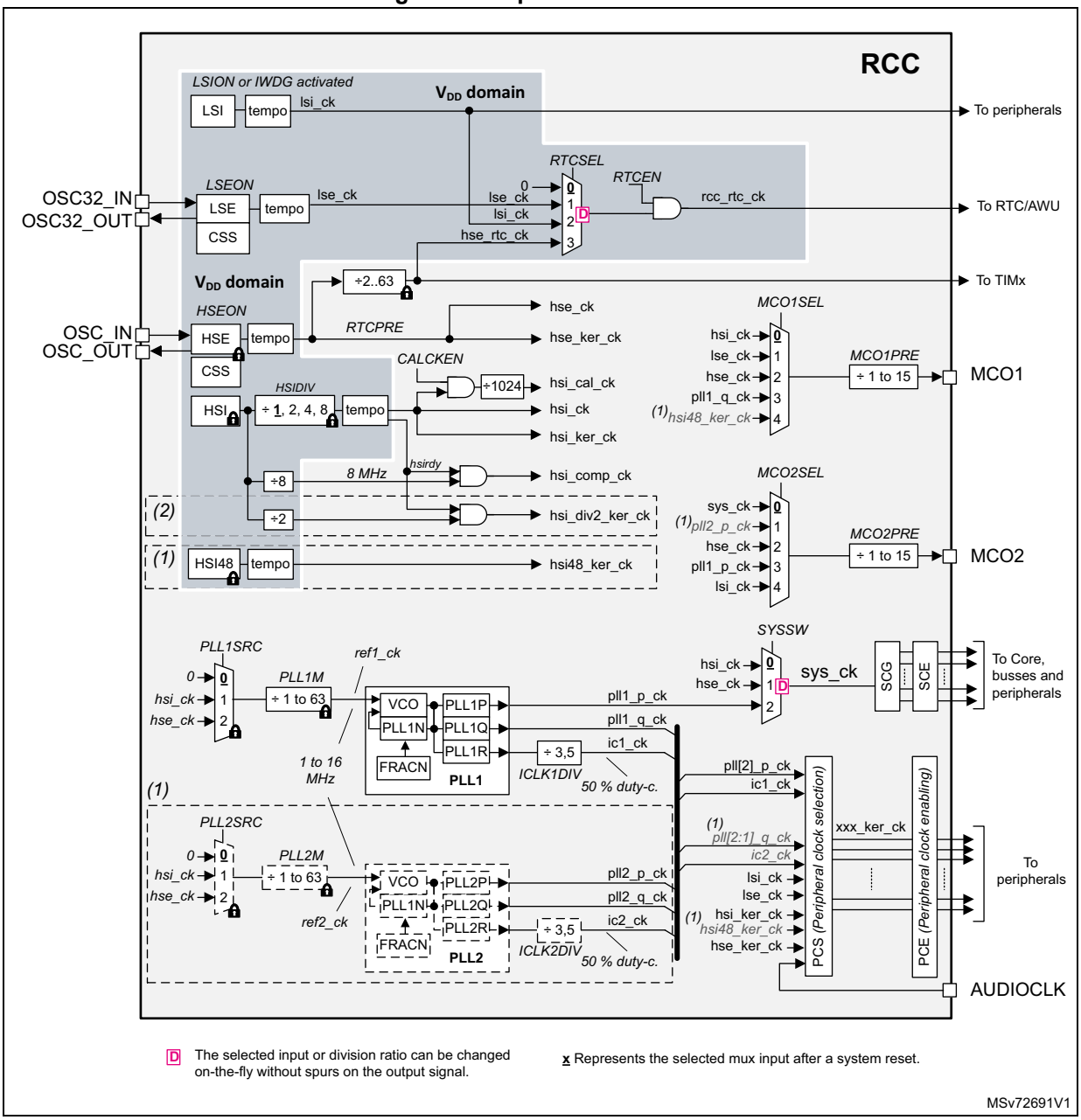
Three different clock sources can be used to drive the system clock (SYSCLK):

- HSI
- HSE
- PLL1

The HSI is used as system clock source after startup from reset, and configured at 64 MHz.

a. PLL2 and HSI48 are not available on the STM32G53x/54x devices. Refer to the device datasheet for availability.

Figure 49. Top-level clock tree



1. The PLL2 and HSI48 are not available on the STM32G53x/54x devices. Refer to the device datasheet for availability.
2. The hsi_div2_ker_ck is not implemented on STM32G55x/56x.

11.5.1 Clock naming convention

The RCC provides the clocks to the whole device. The following terms are used in this document:

- **Peripheral clocks**

  The peripheral clocks are the clocks provided by the RCC to the peripherals. Two kinds of clocks are available:

  – Bus interface clocks

  – Kernel clocks

  A peripheral receives a bus interface clock from the RCC, to access its registers and thus control the peripheral operation. This clock is generally the AHB or APB clock depending on which bus the peripheral is connected to. Some peripherals only need a bus interface clock (such as RNG or TIMx).

  Some peripherals also require a dedicated clock to handle the interface function. These clocks are generally asynchronous with respect to the bus interface clock. They are named kernel clocks (rcc_perx_ker_ck). As an example, peripherals such as SPI in I2S mode must generate specific and accurate synchronization signals, which require dedicated kernel clock frequencies. Another advantage of decoupling the bus interface clock from the specific interface needs, is that the bus clock can be changed without reprogramming the peripheral.

- **System clock**

  The system clock (**sys_ck**) is the root clock for the CPU clock (**cpu_ck**), the bus matrix clock (**bus_ck**), and the peripheral system.

- **Bus matrix clocks**

  Bus matrix clocks are the clocks provided to the different bridges (APB or AHB). These clocks are derived from the main bus matrix clock (**bus_ck**).

11.5.2 Oscillators state summary

Table 115 shows the oscillator states versus system modes, when the oscillators are enabled via registers. The term “available” means that the resource can be used if enabled via registers.

Table 115. Oscillator states versus system modes

Situations	VDD domain				RTC domain
	HSE	HSI	HSI48 (1)	LSI	LSE
After a power-on reset (pwr_por_rst)	OFF	ON	OFF	OFF	OFF
After a system reset (sys_rst)	OFF	ON	OFF	Available	Available
After a system Stop	OFF	ON	OFF	Available	Available
In Run/Sleep	Available	Available	Available	Available	Available
In Stop	OFF	Available (2)	OFF	Available	Available
In Standby	OFF	OFF	OFF	Available	Available
After a RTCDRST reset	Available	Available	Available	OFF	OFF

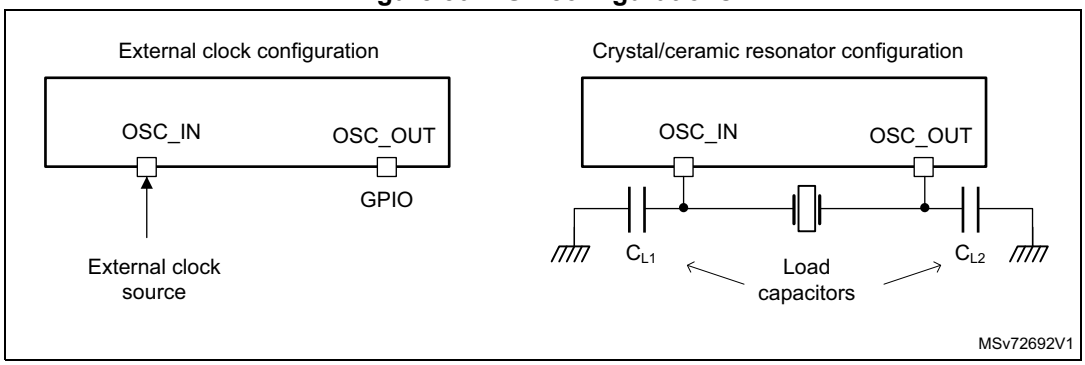
1. The HSI48 is not available on the STM32G53x/54x devices. Refer to the device datasheet for availability.
2. The HSI can remain activated in Stop mode if HSIKERON = 1, or if a peripheral selecting the HSI generates a kernel clock request.

11.5.3 HSE oscillator

The HSE block enables the application to provide a very accurate high-speed clock for the product. The HSE can generate an internal clock from two possible sources:

- External clock source
- External crystal/ceramic resonator

Figure 50. HSE configurations



External clock source (HSE bypass)

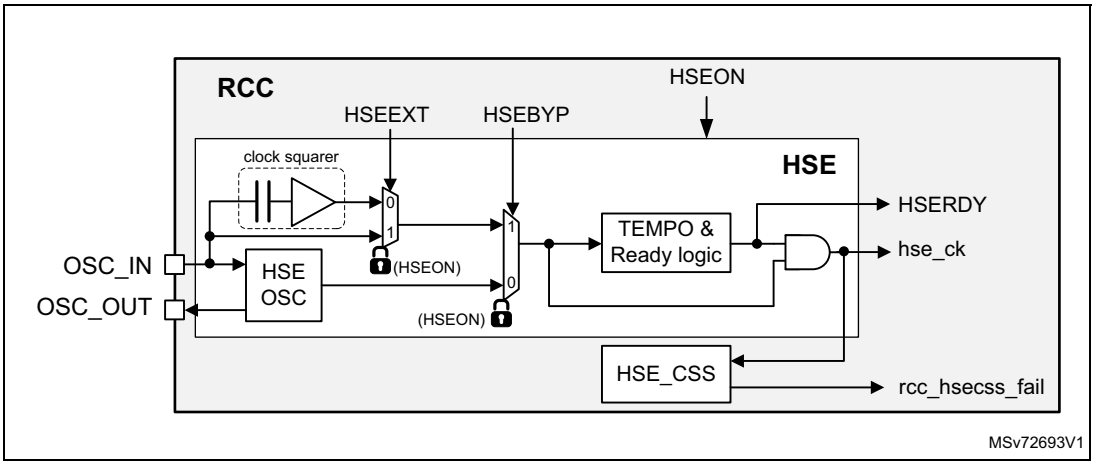
In this mode, the oscillator is not used and an external clock source must be provided to the OSC_IN pin. The external clock can be low swing (analog) or digital. The OSC_OUT pin can be used as GPIO (see *Figure 50*).

The external clock signal can be digital or analog (square, sinus, or triangle). An analog clock signal with a reduced amplitude is supported thanks to an internal clock squarer. The input signal must have a duty cycle close to 50%. Refer to the datasheet for additional information.

This mode is selected when both the HSEBYP and HSEON bits are set. In case of an analog clock input (low swing), the HSEEXT must be cleared. For a digital clock input, the HSEEXT bits must be set.

The HSEEXT, HSEBYP, and HSEON bits are located in the *RCC clock control register (RCC_CR)*.

Figure 51. HSE clock generation



External crystal/ceramic resonator

A crystal/resonator can be connected as shown in *Figure 50*: the crystal/resonator and the load capacitors must be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The oscillator mode is enabled by clearing the HSEBYP bit and setting the HSEON bit.

HSE ready logic

The HSERDY flag indicates when a valid clock is available on the HSE output (**hse_ck**). When the HSE is enabled (HSEON set), the HSERDY flag is set when 256 valid cycles of HSE have been detected. The **hse_ck** clock remains gated until HSERDY is set.

An interrupt can be generated if enabled in the *RCC interrupt enable register (RCC_IER)*.

HSE controls

The HSE can be switched ON and OFF through the HSEON bit.

The hardware does not allow clearing the HSEON bit if one of the following conditions is met:

- The HSE is used directly (via the SYSSW switch) as system clock.
- The HSE is selected as the reference clock for PLL1, with PLL1 enabled and selected to provide the system clock.

The hardware does not allow changing the values of HSEBYP and HSEEXT when HSEON is set.

The HSE is automatically disabled by hardware when the system enters is system Stop or Standby mode (see *Table 115: Oscillator states versus system modes*).

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as clock source for other application components.

Programming sequence

To initialize the HSE, the application must follow the sequence below:

1. Make sure that the HSE is not directly or indirectly used as system clock. If it is the case, use the HSI as clock source for the system clock.
2. Disable the HSE by clearing the HSEON bit.
3. Check that the HSE is disabled by waiting until the bit HSERDY is cleared.
4. If the oscillator mode is needed, select the oscillator mode by clearing HSEBYP.
5. If an external clock is connected to OSC_IN:
   - Select the bypass mode by setting HSEBYP.
   - If the input clock is a full-swing digital signal, set HSEEXT.
   - If the input clock is a low-swing signal, clear HSEEXT.
6. Enable again the HSE by setting the HSEON bit.
7. Wait until HSERDY is set. The HSE is then ready for use.

11.5.4 HSI oscillator

The HSI block provides the default clock to the product.

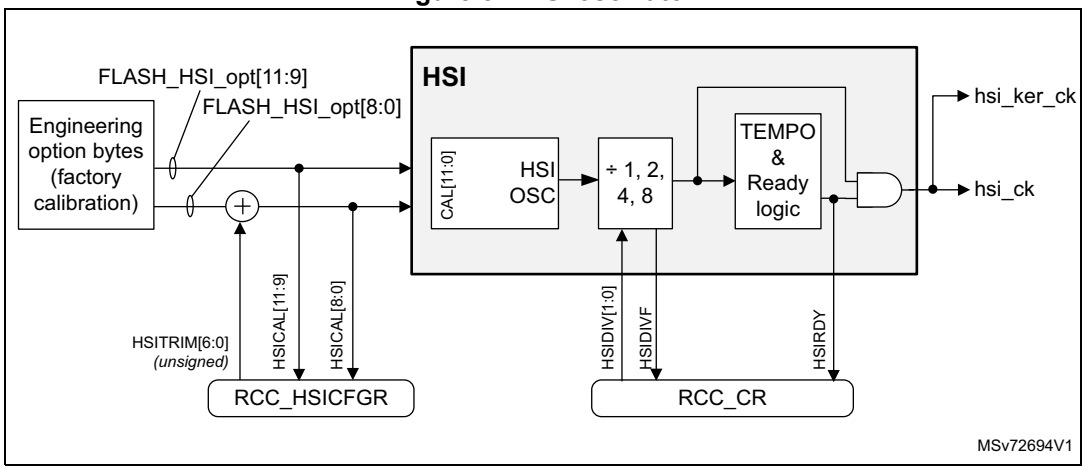
The HSI is a high-speed internal RC oscillator that can be used directly as system clock, peripheral clock, or as PLL input. A predivider enables the application to select an HSI output frequency of 8, 16, 32 or 64 MHz.

The HSI advantages are the following:

- Low-cost clock source, as no external crystal is required
- Faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

Figure 52. HSI oscillator



HSI controls

The HSI can be switched ON and OFF using the HSION bit.

The hardware does not allow clearing the HSION bit if one of the following conditions is met:

- The HSI is used directly (via the SYSSW switch) as system clock.
- The HSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via the SYSSW switch).

In addition, the HSIDIV cannot be changed if the HSI is selected as reference clock for a PLL. In that case, the hardware does not update the HSIDIV with the new value. However, it is possible to change the HSIDIV on-the-fly if the HSI is not used as PLL reference clock. The HSIDIVF flag indicates when the new division ratio is applied.

The HSIRDY flag indicates when a valid clock is available at HSI output. When the HSI is enabled (HSION set), the HSIRDY flag is set when 16 valid cycles provided by HSIDIV have been detected. The HSIDIV output remains gated until HSIRDY is set.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to *Section : Clock security system on HSE*).

The HSI can be disabled or not when the system enters system Stop mode. It is not available in system Standby (see *Table 115: Oscillator states versus system modes*). When exiting Standby mode, the user trim is lost.

In addition, the HSI clock can be driven to the MCO1 output and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals. In this case, the application must consider the following parameters:

- The time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- The frequency accuracy.

HSION, HSIRDY, HSIDIVF, and HSIDIV[1:0] bits are located in the *RCC clock control register (RCC_CR)*.

HSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. This is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACC_{HSI} (refer to the product datasheet for more information).

After a power-on reset, the factory calibration value is loaded into the HSICAL[11:0] bits. If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using the HSITRIM[6:0] bits.

Note: HSICAL[11:0] and HSITRIM[6:0] are located in the RCC HSI calibration register (RCC_HSICFGR).

11.5.5 HSI48 oscillator

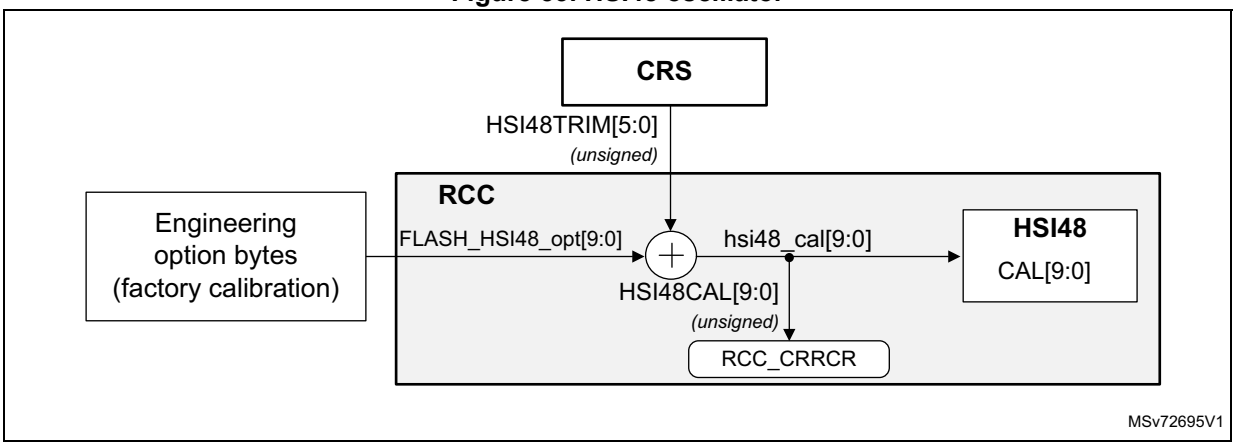
The HSI48 block is not available on the STM32G53x/54x devices. Refer to the device datasheet for availability.

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator, and can be used directly for the USB and the random number generator (RNG).

The internal 48 MHz RC oscillator is mainly dedicated to provide a high-precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE, or an external signal to adjust automatically and quickly the oscillator frequency on-the-fly, with a very small granularity. The calibration value provided to the HSI48 can be read via the *RCC clock recovery RC register (RCC_CRRCR)*

The HSI48TRIM value can be adjusted through the CRS block, for more details, refer to *Section 12: Clock recovery system (CRS)*.

Figure 53. HSI48 oscillator



The HSI48 is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, the HSI48 oscillator runs on its default frequency, which is subject to manufacturing process variations.

The HSI48RDY flag in the *RCC clock control register (RCC_CR)* indicates whether the HSI48 RC oscillator is stable or not. At startup, the HSI48 RC oscillator output clock is not released until this bit is set by hardware.

The HSI48 can be switched ON and OFF using the HSI48ON bit of the *RCC clock control register (RCC_CR)*.

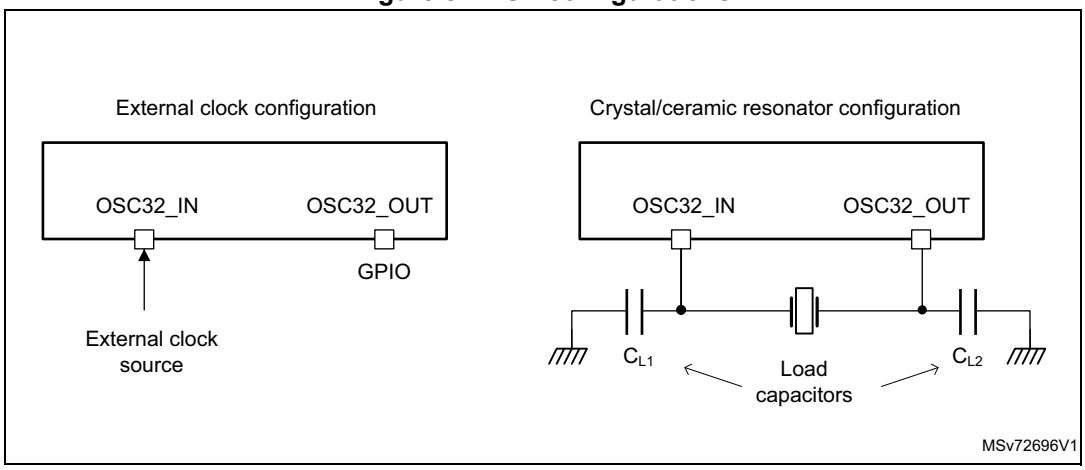
In addition, the HSI48 clock can be driven to the MCO1 output and used as clock source for other application components.

11.5.6 LSE oscillator

The LSE block enables the application to provide a very accurate low-frequency clock for the product. The LSE can generate an internal clock from two possible sources:

- External user clock
- External crystal/ceramic resonator

Figure 54. LSE configurations



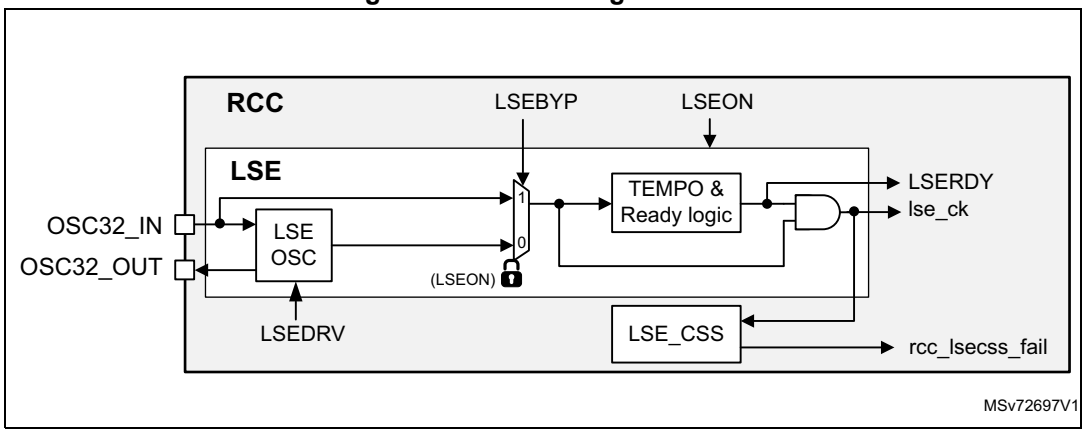
External clock source (LSE bypass)

In this mode, an external clock source must be provided to the OSC32_IN pin. The input clock can have a frequency of up to 1 MHz, and must be a digital signal. A duty cycle close to 50% is recommended.

This external clock is provided to the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO.

The external clock mode is selected by setting the LSEBYP and LSEON bits.

Figure 55. LSE clock generation



External crystal/ceramic resonator (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It provides a low-power highly accurate clock source to the real-time clock (RTC) for clock/calendar or other timing functions. A crystal/resonator can be connected as shown in *Figure 54*: the crystal/resonator and the load capacitors must be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The oscillator mode is selected by clearing the LSEBYP bit and setting the LSEON bit.

The LSE also offers a programmable driving capability (LSEDRV[1:0]) that can be used to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to ensure a stable oscillation. The driving capability must be set before enabling the LSE oscillator.

Warning: It is not allowed to change the driving capability when the LSE is enabled, otherwise the LSE behavior is not guaranteed.

LSE ready logic

The LSE offers an LSERDY flag (see *RCC RTC domain control register (RCC_RTCCR)*), which indicates whether the LSE clock is available or not. When the LSE is enabled (LSEON set), the LSERDY flag is set when a certain number of valid cycles of LSE clock have been detected. The **lse_ck** clock is not released until LSERDY is set.

When LSEBYP = 0, the RCC waits for 4096 clock cycles before activating the LSERDY flag.

When LSEBYP = 1, the RCC waits for one clock cycle.

An interrupt can be generated if enabled in the *RCC interrupt enable register (RCC_IER)*.

LSE controls

The LSE control bits LSEBYP, LSEDRV[1:0], and LSEON are located in the *RCC RTC domain control register (RCC_RTCCR)*. This register is write-protected via the DRTCP bit of the PWR_RTCCR register. To modify the *RCC RTC domain control register (RCC_RTCCR)*, the bit DRTCP must be set.

The LSE oscillator is switched ON and OFF using the LSEON bit. The hardware does not allow changing the LSEBYP values when LSEON is set.

The LSE remains enabled when the system enters Stop or Standby (see *Table 115: Oscillator states versus system modes*).

In addition, the LSE clock can be driven to the MCO1 output and used as clock source for external components.

LSE programming sequence

To initialize the LSE, the application must follow the sequence below:

1. Set the DRTCP bit of the PWR_RTCCR to allow write access to the *RCC RTC domain control register (RCC_RTCCR)*.
2. Disable the LSE by clearing the LSEON bit.
3. Check that the LSE is disabled by waiting until the LSERDY bit is cleared.
4. If the oscillator mode is needed:
   a) Select the oscillator mode by clearing LSEBYP.
   b) If needed, configure LSEDRV[1:0].
5. If an external clock is connected to OSC32_IN, select the bypass mode by setting LSEBYP.
6. Enable again the LSE by setting the LSEON bit.
7. Wait until LSERDY is set. The LSE is then ready for use.
8. If no other protected bits need to be changed, clear the DRTCP bit to write-protect the *RCC RTC domain control register (RCC_RTCCR)*.

11.5.7 LSI oscillator

The LSI acts as a low-power clock source that can be kept running when the system is in Stop or Standby mode for the independent watchdog (IWDG), RTC, and auto-wake-up unit (AWU).

The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the device datasheet.

The LSI can be switched ON and OFF using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced on and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode.

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the *RCC interrupt enable register (RCC_IER)*.

In addition, the LSI clock can be driven to the MCO2 output, and used as a clock source for other application components.

Note: The LSION and LSIRDY bits are located in the RCC RTC domain control register (RCC_RTCCR).

11.5.8 Clock security system (CSS)

The RCC offers a clock security system for the HSE and LSE oscillators. The clock security systems can detect a failure on HSE and LSE oscillators. *Figure 56* shows how this function interacts with other blocks.

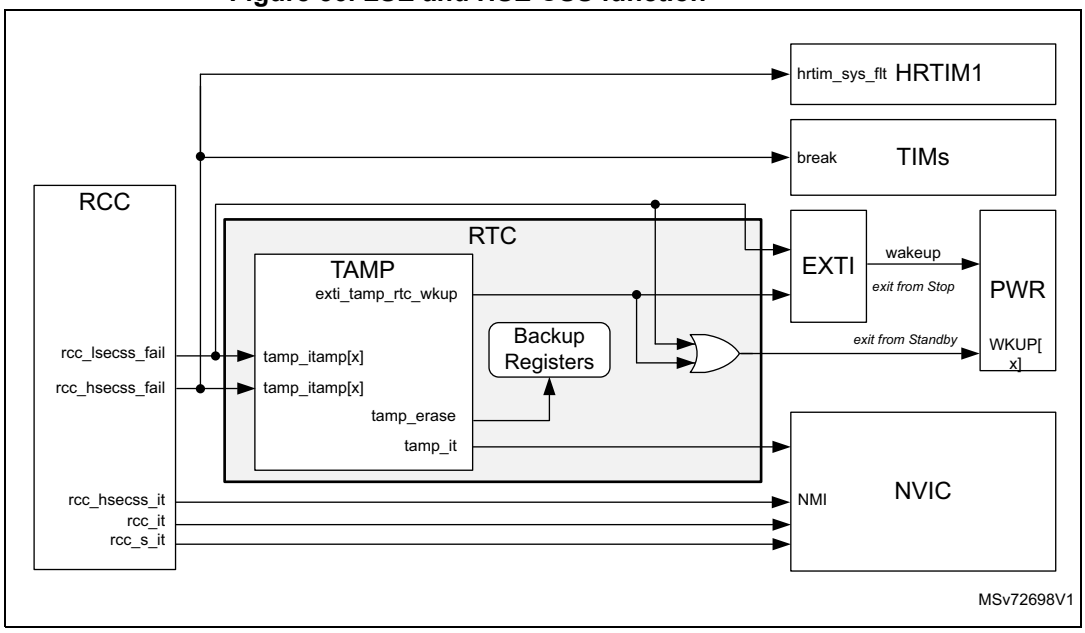
The rcc_lsecss_fail signal is used for several purposes:

- As an input to the TAMP block to generate a tamper event
- As an input of the EXTI, to wake up the system from Stop in case of LSE failure
- As an input to the PWR, to exit the system from Standby in case of LSE failure.

rcc_lsecss_fail is released when LSECSSD is cleared.

The rcc_hsecss_fail signal is routed to some TIMs and to HRTIM1 as break input signal.

Figure 56. LSE and HSE CSS function



Clock security system on HSE

The clock security system on HSE detects when a failure is occurs on the HSE. The clock security system on HSE can be enabled by software via the HSECSSON bit, even when the HSEON is cleared.

The CSS on HSE is activated when the HSE is enabled and ready, and the software sets the HSECSSON bit. As a consequence, the CSS on HSE is not active when the HSE is disabled. For example, this function is not active when the system is in Stop mode.

It is not possible to clear directly the HSECSSON bit by software. The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters Standby mode.

When the CSS on HSE is enabled, the following actions are performed by the RCC if a failure is detected:

1. The signal **rcc_hsecss_fail** is asserted.
2. The HSI clock is enabled (HSION set) and the SYSSW switch automatically selects HSI as system clock.
3. The HSE is then automatically disabled (HSEON is cleared).
4. If the HSE output was used as clock source for PLLs, the PLLs are also disabled.
5. The clock failure event (**rcc_hsecss_fail**) is also sent to the break inputs of advanced-control timers and the hrtim_sys_flt input of high-resolution timer (HRTIM1).
6. An NMI interrupt is generated to inform the software about the failure (CSS interrupt: ***rcc_hsecss_it***), thus allowing the MCU to perform rescue operations. The CSS interrupt is asserted until the HSECSSF bit is cleared. The HSECSSF flag can be cleared by setting the HSECSSC bit in the *RCC interrupt flag register (RCC_IFR)*.

A tamper event can also be triggered to clear the content of the backup registers.

Clock security system on LSE

The clock security system on LSE detects when a failure is occurs on the LSE.The clock security system on the LSE oscillator can be enabled by programming the LSECSSON bit in the *RCC RTC domain control register (RCC_RTCCR)*.

This bit is disabled by hardware if one of the following conditions is met:

- After a power-on hardware reset (**pwr_por_rst)**
- After an RTC domain software reset via RTCDRST bit

The software can also disable the clock security system after an LSE failure detection.

The CSS on LSE works in Run, Stop, and Standby.

The LSECSS offers two flag signals:

- The LSECSSD, capable of retaining an LSE failure even in Standby mode.
- The LSECSSF, used to generate an interrupt in case of LSE failure.

The sequence below describes the LSE enabling sequence with the clock security system enabled:

1. Follow the LSE enable procedure given is the section *Section : LSE programming sequence*, except for the last step.
2. Select the LSE clock via RTCSEL[1:0].
3. Set the LSECSSON bit.
4. If no other protected bits need to be changed, clear the DRTCP bit to write-protect the *RCC RTC domain control register (RCC_RTCCR)*.

Note: Set the LSECSSON bit only after the LSE is enabled and ready, and after the RTC clock has been selected through the RTCSEL bits.

If a failure is detected on the LSE, the hardware performs the following actions:

1. The LSE clock is no more delivered to the RTC.
2. LSERDY is cleared, but the values of RTCSEL, LSECSSON and LSEON bits are not changed by the hardware.
3. A failure event is generated (**rcc_lsecss_fail**). This event enables the system to wake up from Stop or Standby, but also protects the backup registers via the TAMP block.
4. The interrupt flag LSECSSF is set to generate an interrupt (**rcc_lsecss_it**).
5. The LSECSSD flag is also set, retaining the first LSE failure even if it occurred in Standby.

On the software side, different actions can be taken according to the application requirements:

- If the application no longer intends to use LSE when a failure has been detected:
  a) Unlock the *RCC RTC domain control register (RCC_RTCCR)* by setting the DRTCP bit of PWR_RTCCR.
  b) Clear the LSECSSF bit, if an interrupt was enabled for this event.
  c) Disable the LSE by clearing the LSEON bit.
  d) If needed, change the clock source for the RTC:
  – Clear the RTCEN bit to disable the RTC clock.
  – Enable the new clock source for the RTC.
  – Select the proper clock source via RTCSEL.
  – Set the RTCEN bit to enable the RTC clock.
  e) Clear the LSECSSON bit.
  f) The application must perform specific actions consecutive to TAMPER events if enabled.
  g) If no other protected bits need to be changed, clear the bit DRTCP to write-protect the *RCC RTC domain control register (RCC_RTCCR)*.
- If the application wants to reinitialize the RTC domain:
  a) Unlock the *RCC RTC domain control register (RCC_RTCCR)* by setting the DRTCP bit of the PWR_RTCCR.
  b) Perform an RTC domain reset by setting the RTCDRST bit, then clear it.
  c) The application must perform specific actions consecutive to TAMPER events if enabled.
  d) Reinitialize all components of the RTC domain.
  e) If no other protected bits need to be changed, clear the DRTCP bit to write-protect *RCC RTC domain control register (RCC_RTCCR)*.

- If the application wants to reuse the LSE when a failure has been detected:
  a) Unlock the *RCC RTC domain control register (RCC_RTCCR)* by setting the DRTCP bit of the PWR_RTCCR.
  b) Clear the LSECSSF bit if an interrupt was enabled for this event.
  c) The application must perform specific actions consecutive to TAMPER events if enabled.
  d) Clear the LSECSSON bit. This action also clears the LSECSSD flag.
  e) Set LSECSSON again.
  f) Wait until LSERDY is set. The LSERDY flag is set after the oscillator settling time delay plus 4096 LSE clock periods. If it is not the case, it probably means that th LSE failure is permanent.

Refer to the device datasheet for CSS on LSE electrical characteristics.

11.5.9 Clock output generation (MCO1, MCO2)

Two microcontroller clock output pins (MCO1 and MCO2) are available. A clock source can be selected for each output. The selected clock can be divided thanks to a configurable prescaler.

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1[2:0], MCO2PRE[3:0], and MCO2[2:0], located in the *RCC clock configuration register1 (RCC_CFGR1)*.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

The dividers MCO1PRE and MCO2PRE provide a clock with a duty cycle of 50% for even division values. More generally the duty cycle is given by the following formula:

$$DC(\%) = \frac{\text{FLOOR}\left(\frac{\text{MCOxPRE}}{2}\right)}{\text{MCOxPRE}} \times 100$$

Where MCOxPRE = 2 to 15.

The clock provided to the MCOs outputs must not exceed the maximum pin speed (refer to the device datasheet for information on the supported speed).

MCO1 and MCO2 outputs are available in Stop mode, but not in Standby mode.

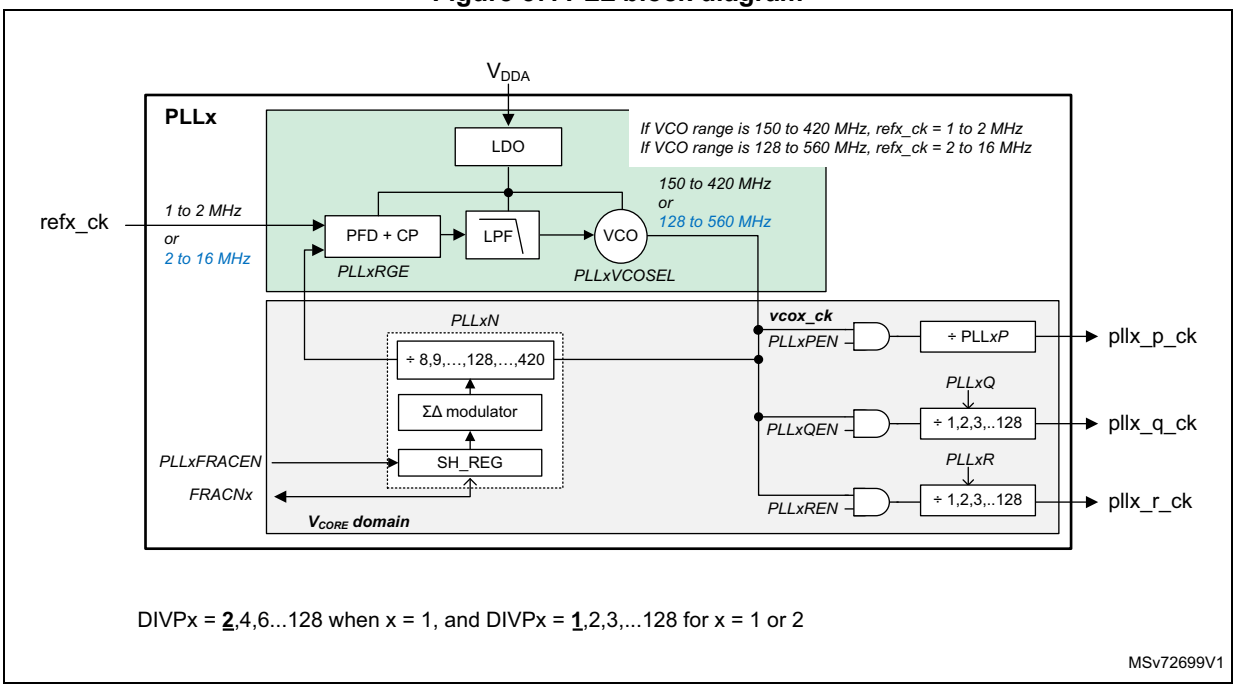
11.5.10 PLL description

The STM32G56x/55x/54x/53x devices feature up to two PLLs, PLL1 and PLL2. PLL2 is only available on STM32G55x/56x (refer to the device datasheet for details).

The PLLs integrated into the RCC are completely independent. They offer the following features:

- A VCO supporting two modes:
  - A wide-range mode (128 to 560 MHz)
  - A low-range used mode (150 to 420 MHz)
- Input frequency range:
  - 2 to 16 MHz for the VCO in wide-range mode
  - 1 to 2 MHz for the VCO in low-range mode
- Capability to work either in integer or fractional mode
- 13-bit sigma-delta modulator.
- The sigma-delta modulator can be updated on-the-fly without generating frequency overshoots on PLLs outputs
- Each PLL offers three outputs with postdividers

Figure 57. PLL block diagram



The PLLs are controlled via the RCC_PLLxDIVR, RCC_PLLxFRACR, RCC_PLLCFGR, and RCC_CR registers.

The frequency of the reference clock provided to the PLLs (**refx_ck**) must range from 1 to 16 MHz. The PLLxM dividers of the RCC_PLLxCFGR register must be properly programmed to match this condition. In addition, the PLLxRGE[1:0] bitfield of the RCC_PLLxCFGR registers must be set according to the reference input frequency, to optimize performance.

The user application can then configure the VCO. The smaller range must be chosen when the reference clock frequency is lower than 2 MHz. To reduce the power consumption, it is recommended to configure the VCO output to the smaller range.

DIVNx loop divider must be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range must be respected.

The PLLs operate in integer mode when the value of the SH_REG bit of the FRACNx shadow register is cleared. The SH_REG bit is updated with the FRACNx value when the PLLxFRACEN bit goes from 0 to 1. The sigma-delta modulator is designed to minimize the jitter impact, while allowing very small frequency steps.

The PLLs can be enabled by setting PLLxON. The PLLxRDY bits indicate that the PLL is ready (locked).

*Note: Before enabling the PLLs, make sure that the reference frequency (**refx_ck**) is stable, since PLLxM cannot be changed when PLLx is ON, neither can PLLSRC when one of the PLLs is ON.*

The hardware prevents clearing PLL1ON if the PLL1 is currently used to deliver the system clock. There are other hardware protections on the clock generators (refer to HSE oscillator, HSI oscillator, and HSI48 oscillator).

The following PLL parameters cannot be changed once the PLL is enabled: DIVNx, PLLxRGE, PLLxVCOSEL, PLLxP (DIVP), PLLxQ (DIVQ), and PLLxR (DIVR).

For optimal behavior of the PLL when one of the postdividers (DIVP, DIVQ, or DIVR) is not used, the application must clear the enable bit (DIVyEN) as well as the corresponding postdivider bits (DIVP, DIVQ, or DIVR).

If the above rules are not respected, the PLL output frequency is not guaranteed.

PLLs frequency limits

The next table shows the most important frequency limits the application must respect when programming the PLLs.

Table 116. PLLs frequency limits

Blocks	Functional signals	Frequency range(1)			Applicable voltage scaling	Comments
		Min	Max	Unit		
PLL1, PLL2(2)	pllx_p_ck	-	F_{MAX}	MHz	All VOS	PLLxP output
	pllx_q_ck	-	F_{MAX}		All VOS	PLLxQ output
	pllx_r_ck	-	500		Only VOS0	PLLxR output, connected to ICLKxDIV
		-	400		All VOS	
	vcox_ck	128	560		All VOS	PLLxVCOSEL = 0 and PLLxRGE = 1, 2 or 3
	refx_ck	2	16		All VOS	
	vcox_ck	150	420		All VOS	PLLxVCOSEL = 1 and PLLxRGE = 0
	refx_ck	1	2		All VOS	

1. F_{MAX} value represents the maximum frequency at which the CPU can run. This value depends on the STM32G56x/55x/54x/53x device and can be found on the device datasheet.
2. The PLL2 is not available on the STM32G53x/54x devices. Refer to the device datasheet for availability.

Output frequency computation

When the PLL is configured in integer mode (SH_REG = 0), the VCO frequency (F_{VCO}) is given by the following formulas:

$$F_{VCO} = F_{REF_CK} \times DIVN$$

$$F_{PLL_y_CK} = (F_{VCO} / (DIVy + 1)) \text{ with y = P, Q, or R}$$

When the PLL is configured in fractional mode (SH_REG different from 0), the DIVN divider must be initialized before enabling the PLLs. However, it is possible to change the value of FRACNx on-the-fly without disturbing the PLL output.This feature can be used either to generate a specific frequency from any crystal value with a good accuracy, or to fine-tune the frequency on-the-fly.

For each PLL, the VCO frequency is given by the following formula:

$$F_{VCO} = F_{ref_ck} \times \left(DIVN + \frac{FRACN}{2^{(13)}} \right)$$

Note: For PLL1, DIVP can only take odd values.

The PLLs are disabled by hardware when:

- The system enters Stop or Standby mode.
- An HSE failure occurs when HSE or PLL (clocked by HSE) are used as system clock.

Recommended PLL initialization sequence

The PLLx are supposed to be disabled at the start of the initialization sequence:

1. Enable the oscillator used as PLL clock source, and wait for oscillator ready state.
2. Select the clock source via PLLxSRC[1:0].
3. Select the predivider ratio via PLLxM[5:0].
4. Configure the PLLx:
   a) Select the correct input frequency range via PLLxRGE[1:0] and the VCO range via PLLxVCOSEL.
   b) Set the VCO frequency via PLLxN[8:0].
   c) If PLLx is used in integer mode:
      Set PLLxFRACEN.
      Clear PLLxFRACN[12:0].
      Clear PLLxFRACEN.
   d) If PLLx is used in fractional mode:
      Set the fractional multiplication ratio into PLLxFRACN[12:0].
      Set PLLxFRACEN.
   e) Select the division ratio for each PLL output: PLLxP[6:0], PLLxQ[6:0], and PLLxR[6:0].
   f) Enable DIVP, DIVQ, and DIVR if used (PLLxPEN, PLLxQEN, PLLxREN).
5. Enable the PLLx by setting PLLxON.

6. Wait until PLLxRDY = 1. If the PLLx is in fractional mode, the PLLxFRACEN bit must not be cleared again as long as PLLxRDY = 0.
7. If ICLK1 or ICLK2 must be enabled, just enable the corresponding PLLxREN.
8. PLLx is ready to use.

Note: When the PLLxRDY is set, the difference between the PLLx output frequency and the target value is lower than ±2%.

Updating the fractional value

The fractional part of the PLLx can be updated at any time by following the sequence below:

1. Clear PLLxFRACEN.

   When PLLxFRACEN = 0, the sigma-delta modulator still operates with the current PLLxFRACN[12:0] value. The application must wait for three clock periods of refx_ck (PLLxFRACEN bit propagation delay).
2. Write a new fractional value to PLLxFRACN[12:0].
3. Set PLLxFRACEN to take into account the new PLLxFRACN value. The new value is taken into account after three clock periods of refx_ck (PLLxFRACEN bit propagation delay).

Note: The disabling of ICLK1 and ICLK2 can be simply done by clearing PLL1REN or PLL2REN.

Changing the division ratio of ICLK1 or ICLK2 can be done on-the-fly, without disabling ICLK1 or ICLK2.

11.5.11 Intermediate clock generation (ICLKx)

The STM32G55x/56x devices offer two PLLs, while the STM32G53x/54x feature only one. As a consequence, the former feature both ICLK1 and ICLK2, while the latter only features ICLK1 (refer to the device datasheet for availability).

The DIVR outputs of PLL1 and PLL2 are directly connected to intermediate clock dividers.

Each intermediate clock divider allows division by 3 and by 5, to improve the duty cycle and match the needs of some peripherals.

Changing the division ratio can be done on-the-fly, without disabling the ICLKx. The disabling of ICLK1 and ICLK2 can be simply done by clearing PLL1REN or PLL2REN.

ICLK1DIV and ICLK2DIV are located in the *RCC PLL clock source selection register (RCC_PLL1CFGR)* and *RCC PLL clock source selection register (RCC_PLL2CFGR)*, respectively.

Note that the duty cycle at ICLKx output depends on the duty cycle of the input clock. *Table 117* gives some value examples.

Table 117. Resulting duty cycle examples

Duty cycle of the input clock	Division ratio	Duty cycle of the ICLKx output clock
33 %	3	44.3 %
	5	46.6 %
40 %	3	46.7 %
	5	48.0 %

Table 117. Resulting duty cycle examples (continued)

Duty cycle of the input clock	Division ratio	Duty cycle of the ICLKx output clock
45 %	3	48.3 %
	5	49.0 %

Note: If DIVR selects even division factors, the duty cycle at ICLKx output is 50%.

11.5.12 System clock (sys_ck)

System clock selection

When the system is running, the user application can select the system clock (**sys_ck**) among the following sources:

- HSI
- HSE
- pll1_p_ck

After a system reset (or after exiting Standby mode), the HSI is selected as system clock and all PLLs are switched OFF. When a clock source is used for the system clock, the software cannot disable the selected source via the xxxON bits.

The system clock can be stopped by the hardware when the system enters Stop or Standby mode.

The system switch function is controlled by programming the SYSSW[1:0]. A switch from one clock source to another occurs only if the targeted clock source is enabled and ready (clock stable after startup delay or PLL ready). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready.

The status bits SYSSWS[1:0] of the *RCC clock configuration register1 (RCC_CFGR1)* indicate which clock is currently used as a system clock.

System clock generation

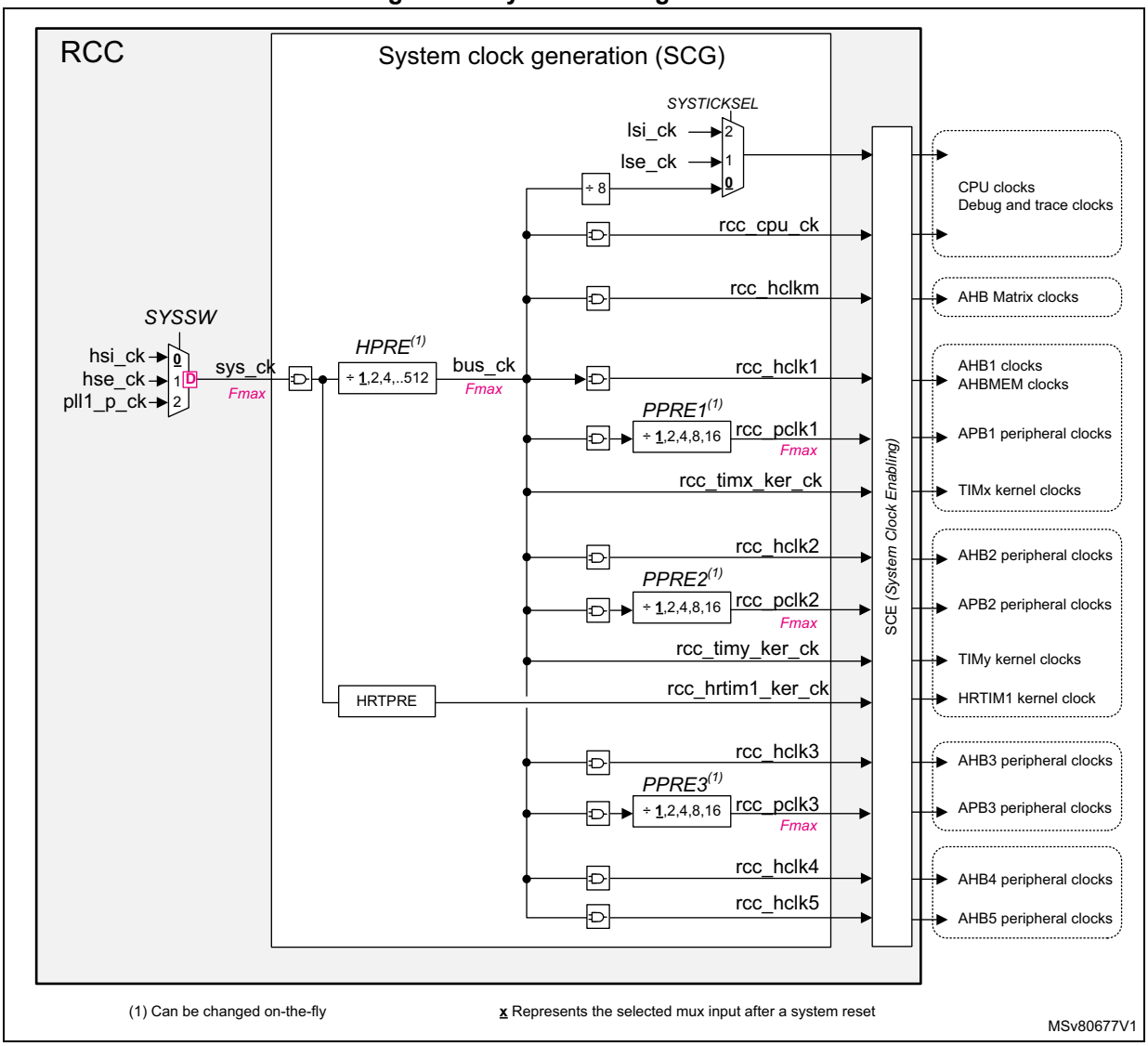
Figure 58 shows a simplified view of the clock distribution for the CPUs and buses. All the dividers shown in the block diagram can be changed on-the-fly without generating timing violations. This feature is a very simple solution to adapt the bus frequencies to the application needs, thus optimizing the power consumption.

When the system clock (sys_ck) is provided by the PLL1, it is possible to change dynamically the core frequency by simply updating the DIVP division ratio. Even divisions are preferred. Odd divisions can be used with care: if F_{MAX} is the maximum allowed frequency for the CPU, the high and low level of the CPU clock must not be shorter than $1 / (2 \times F_{MAX})$. Refer to *Section 11.5.10: PLL description* for details.

Changing the DIVP value of PLL1, impacts the CPUs clock frequency but also the clock frequency of all bus matrices.

The CPUs and all AHB matrices are working at the same frequency. PPREx dividers allow to adapt independently the frequency of APB buses.

Figure 58. System clock generation



1. Divider values can be changed on-the-fly. All clocks are phase-aligned.

Note: The application must respect the maximum allowed frequency F_{MAX}. Refer to the product datasheet for maximum values.

*The trace clock (**trace_ck**) is generated from the **cpu_ck** clock. For additional information, refer to Section : Clock distribution for debug and trace.*

11.5.13 Handling clock generators in Stop and Standby modes

Entering Stop mode

When the whole system enters Stop mode, all the clocks (system and kernel clocks) are stopped, as well as the following clock sources:

- HSI
- HSE
- PLL1 and PLL2 if present (refer to the device datasheet for availability)
- HSI48 if present (refer to the device datasheet for availability).

The content of the RCC registers is not altered except for PLLxON, HSEON, and HSI48ON, which are cleared.

If a flash memory programming operation is ongoing, Stop or Standby mode entry is delayed until the flash memory interface access is finished.

If an access to the APB domain is ongoing, Stop or Standby mode entry is delayed until the APB access is finished.

During Stop mode

There are two specific cases where the HSI can be enabled during in Stop mode:

- When a dedicated peripheral requests the kernel clock, the peripheral receives the HSI according to the kernel clock source selected for this peripheral.
- When the HSIKERON bit of the *RCC clock control register (RCC_CR)* is set, the HSI is kept running, but the output is gated. The clock is then available immediately when the system exits Stop mode, or when a peripheral requests the kernel clock (see *Table 118* for details).

Table 118. HSIKERON behavior

HSIKERON	HSI state during Stop mode	HSI state setting time
0	OFF	$t_{su(HSI)}$(1)
1	Running and gated	Immediate

1. $t_{su(HSI)}$ is the startup time of the HSI oscillator (refer to the device datasheet for the value of this parameter).

Caution: HSIKERON must be set only when the HSION is set.

Exiting Stop mode

When the system exits Stop mode via a wake-up event, the device restarts with the HSI oscillator.

Exiting Standby mode

When the system exits Standby mode, the HSI is restarted and selected as system and kernel clock.

The RCC registers are reset to their initial values except for the *RCC reset status register (RCC_RSR)* and *RCC RTC domain control register (RCC_RTCCR)*.

Note: The HSI output provides two clock paths:

- *One path for the system clock (hsi_ck)*
- *One path for the peripheral kernel clock (hsi_ker_ck).*

When a peripheral requests the kernel clock in system Stop mode, only the path providing the hsi_ker_ck is activated.

11.5.14 Peripheral clock distribution

The RCC handles the distribution of the CPU, interconnect, bus interface, and peripheral clocks for the whole device, according to the CPU operating mode.

The clocks used as bus interface for peripherals can be hclkx or pclkx, depending on the bus connected to each peripheral. Some peripherals also request additional clocks (named kernel clocks) to handle specific functions.

The peripheral clock bus interfaces and kernel clocks can be asynchronous or synchronous.

Having separate kernel and peripheral bus clocks enables the user application to change the bus frequency without disturbing the service provided by the peripheral.

For each peripheral, the application can control the activation/deactivation of its kernel and bus interface clock via the RCC block.

Prior to using a peripheral, the application must enable its clocks (by setting PERxEN), and define if this peripheral remains active in low-power mode (by setting PERxLPEN). PERxEN and PERxLPEN bits are located in the RCC_AHBxENR, RCC_APBxENR, RCC_AHBxLPENR and RCC_APBxLPENR registers (PERx is the peripheral name).

When the peripheral clocks are not active, read or write accesses to the peripheral registers are not supported.

When an enable bit is set or when a kernel clock request is asserted, the clock is active after two kernel clock cycles. The peripheral bus interface clock is also affected by this synchronization mechanism.

Caution: Just after enabling the clock for a peripheral, the software must wait for these two clock cycles before accessing the peripheral registers.

Figure 59 shows typical peripheral clock distribution. The PCC (peripheral clock control) block provides a synchronization mechanism (not shown in the figure) that offers a glitch-free clock enabling or disabling.

In addition, the clock switch controlled by PERxSEL and the kernel clock request are not always present (see *Table 119*). This clock switch is glich-free and can be changed on-the-fly. These clock switches and their control are reset by sys_rst unless otherwise specified.

Figure 59. Typical peripheral clock distribution

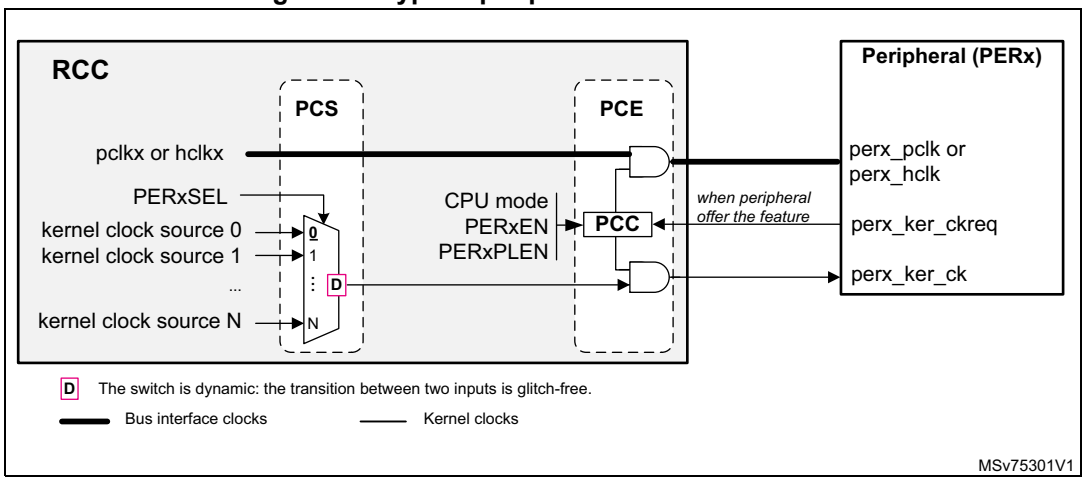


Table 119 shows the peripheral clocks that the RCC can deliver to most of the peripherals. For each peripheral, *Table 119* shows the possible kernel clock sources, the name of the kernel clock selector, the bus clock source, and the maximum allowed frequency.

The gray cells indicate the signals or peripherals that are not present on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet to find out which blocks are present in the product you are using.

Table 119. Peripheral clock distribution summary

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max. freq. (MHz)(1)	Type (2)
			Pos.	Control bitfield		
ADC12	Kernel	rcc_tim_ker_ck	**0**	ADCSEL	80	A
		rcc_hrtim_ker_ck	1			
		hsi_ker_ck	2			
		hse_ker_ck	3			
		pll1_q_ck	4			
		ic1_ck	5			
		pll2_p_ck(3)	6			
		ic2_ck(3)	7			
	Bus	hclk2	-	-	F_{MAX}	-

Table 119. Peripheral clock distribution summary (continued)

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max. freq. (MHz)(1)	Type (2)
			Pos.	Control bitfield		
ADC3(4)	Kernel	rcc_tim_ker_ck	**0**	ADCSEL	80	A
		rcc_hrtim_ker_ck	1			
		hsi_ker_ck	2			
		hse_ker_ck	3			
		pll1_q_ck	4			
		ic1_ck	5			
		pll2_p_ck	6			
		ic2_ck	7			
	Bus	hclk2	-	-	F_{MAX}	-
AES	Bus	hclk4	-	-	F_{MAX}	-
CCB(4)	Bus	hclk4	-	-	F_{MAX}	-
COMP[8:1]	Bus	pclk3	-	-	F_{MAX}	-
CORDIC1	Bus	hclk1	-	-	F_{MAX}	-
CRC1	Bus	hclk1	-	-	F_{MAX}	-
CRS(4)	Bus	pclk1	-	-	F_{MAX}	-
DLYB(4)	Bus	hclk4	-	-	F_{MAX}	-
DAC[2:1]	Kernel	lse_ck	**0**	DACxSHSEL	1	A
		lsi_ck	1			
	Kernel	rcc_tim_ker_ck	**0**	DAC[2:1]SEL	F_{MAX}	A
		rcc_hrtim_ker_ck	1			
		rcc_adc_ker_ck	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		pll1_q_ck	5			
		pll2_q_ck(3)	6			
	Bus	hclk2	-	-	F_{MAX}	-

Table 119. Peripheral clock distribution summary (continued)

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max. freq. (MHz)(1)	Type (2)
			Pos.	Control bitfield		
DAC[10:5]	Kernel	rcc_tim_ker_ck	**0**	DAC[15:0]SEL	F_{MAX}	A
		rcc_hrtim_ker_ck	1			
		rcc_adc_ker_ck	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		pll1_q_ck	5			
		pll2_q_ck(3)	6			
	Bus	hclk2	-	-	F_{MAX}	-
DBGMCU	Bus	hclk3	-	-	F_{MAX}	-
DTS	Kernel	lse_ck	-		1	A
	Bus	pclk1	-		F_{MAX}	
EXTI	Bus	hclk3	-	-	F_{MAX}	-
FLASH1	Bus	hclk1	-	-	F_{MAX}	-
		hclkm		-	F_{MAX}	-
FDCAN1	Kernel	hse_ker_ck	**0**	FDCANSEL	F_{MAX}	A
		pll1_q_ck	1			
		pll2_q_ck (3)	2			
	Bus	pclk1	-	-	F_{MAX}	-
FDCAN2, FDCAN3(4)	Kernel	hse_ker_ck	**0**	FDCANSEL	F_{MAX}	A
		pll1_q_ck	1			
		pll2_q_ck	2			
	Bus	pclk1	-	-	F_{MAX}	-
FMAC1	Bus	hclk1	-	-	F_{MAX}	-
FMAC2 (4)	Bus	hclk2	-	-	F_{MAX}	-
GPDMA1, GPDMA2	Bus	hclk1	-	-	F_{MAX}	-
		hclkm	-	-	F_{MAX}	-
GPIOA-E	Bus	hclk4	-	-	F_{MAX}	-
GPIOF-I(4)	Bus	hclk4	-	-	F_{MAX}	-
HASH	Bus	hclk4	-	-	F_{MAX}	-
HRTIM1	Kernel	rcc_hrtim_ker_ck	-	-	F_{MAX}	S
	Bus	pclk2	-	-	F_{MAX}	-

Table 119. Peripheral clock distribution summary (continued)

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max. freq. (MHz)(1)	Type (2)
			Pos.	Control bitfield		
I2C1, I2C2	Kernel	pclk1	**0**	I2C1SEL, I2C2SEL	F_{MAX}	A
		ic1_ck	1			
		hsi_ker_ck	2			
		hse_ker_ck	3			
	Bus	pclk1	-	-	F_{MAX}	-
I2C3, I2C4(4)	Kernel	pclk3	**0**	I2C3SEL, I2C4SEL	F_{MAX}	A
		ic1_ck	1			
		hsi_ker_ck	2			
		hse_ker_ck	3			
		hsi48_ker_ck	4			
	Bus	pclk3	-	-	F_{MAX}	-
I3C1	Kernel	pclk1	**0**	I3C1SEL	257	A
		ic1_ck	1			
		hsi_ker_ck	2			
		hse_ker_ck	3			
	Bus	pclk1	-	-	F_{MAX}	-
IWDG	Kernel	lsi_ck	-	-	1	A
	Bus	pclk1	-	-	F_{MAX}	-
LPTIM1	Kernel	pclk3	**0**	LPTIM1SEL	F_{MAX}	A
		pll1_q_ck	1			
		pll2_p_ck(3)	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		lsi_ck	5			
		lse_ck	6			
	Bus	pclk3	-	-	F_{MAX}	-
LPUART1	Kernel	pclk3	**0**	LPUART1SEL	F_{MAX}	A
		ic1_ck	1			
		pll2_q_ck(3)	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		lse_ck	5			
	Bus	pclk3	-	-	F_{MAX}	-

Table 119. Peripheral clock distribution summary (continued)

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max. freq. (MHz)(1)	Type (2)
			Pos.	Control bitfield		
XSPI1(4)	Kernel	hclk4	**0**	XSPI1SEL	F_{MAX}	A
		pll1_q_ck	1			
		ic2_ck	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
	Bus	hclk4	-	-	F_{MAX}	-
		hclkm				
OPAMP[3:1](4)	Bus	pclk3	-	-	F_{MAX}	-
OTFDEC1(4)	Bus	hclk4	-	-	F_{MAX}	-
PKA	Bus	hclk4	-	-	F_{MAX}	-
PLAY1(4)	Kernel	lse_ck	**0**	PLAY1SEL	F_{MAX}	A
		lsi_ck	1			
		pll2_p_ck	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		hsi48_ker_ck	5			
	Bus	pclk2	-	-	F_{MAX}	-
POSENC1, POSENC2(4)	Kernel	pll1_p_ck	**0**	POSENCSEL	200	A
		pll1_q_ck	1			
		ic1_ck	2			
		pll2_p_ck	3			
		pll2_q_ck	4			
		ic2_ck	5			
	Bus	pclk3	-	-	F_{MAX}	-
PWR	Bus	hclk3	-	-	F_{MAX}	-
RAMCFG	Bus	hclk1	-	-	F_{MAX}	-
RCC	Bus	hclk3	-	-	F_{MAX}	-
RNG	Kernel	hsi48_ker_ck or hsi_ker_ck / 2(5)	-	-	48	-
	Bus	hclk4	-	-	F_{MAX}	-

Table 119. Peripheral clock distribution summary (continued)

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max. freq. (MHz)(1)	Type (2)
			Pos.	Control bitfield		
RTC/AWU	Kernel	no clock	**0**	RTCSEL	1	A
		lse_ck	1			
		lsi_ck	2			
		hse_ker_ck / (RTCPRE + 1)	3			
	Bus	pclk3	-	-	F_{MAX}	-
SAES (4)	Bus	hclk4	-	-	F_{MAX}	-
SBS	Bus	pclk3	-	-	F_{MAX}	-
SPI1	Kernel	pclk2	**0**	SPI1SEL	F_{MAX}	A
		ic1_ck	1			
		pll2_p_ck(3)	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		AUDIOCLK	5			
	Bus	pclk2	-	-	F_{MAX}	-
SPI2	Kernel	pclk1	**0**	SPI2SEL	F_{MAX}	A
		ic1_ck	1			
		pll2_p_ck (3)	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		AUDIOCLK	5			
	Bus	pclk1	-	-	F_{MAX}	-
SPI3(4)	Kernel	pclk1	**0**	SPI3SEL	F_{MAX}	A
		ic1_ck	1			
		pll2_p_ck	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
	Bus	pclk1	-	-	F_{MAX}	-
TAMP	Kernel	no clock	**0**	RTCSEL	1	A
		lse_ck	1			
		lsi_ck	2			
		hse_ker_ck / (RTCPRE + 1)	3			
	Bus	pclk3	-	-	F_{MAX}	-

Table 119. Peripheral clock distribution summary (continued)

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max. freq. (MHz)(1)	Type (2)
			Pos.	Control bitfield		
TIM3 (4)	Kernel	bus_ck	-	-	F_{MAX}	S
	Bus	pclk1	-	-	F_{MAX}	-
TIM2, TIM6, TIM7, TIM9, TIM10, TIM11, TIM12, TIM13, TIM14	Kernel	bus_ck	-	-	F_{MAX}	S
	Bus	pclk1	-	-	F_{MAX}	-
TIM1, TIM8, TIM15, TIM16, TIM17,	Kernel	bus_ck	-	-	F_{MAX}	S
	Bus	pclk2	-	-	F_{MAX}	-
TIM5, TIM4 (4)	Kernel	bus_ck	-	-	F_{MAX}	S
	Bus	pclk2	-	-	F_{MAX}	-
TIM20 (4)	Kernel	bus_ck	-	-	F_{MAX}	S
	Bus	pclk3	-	-	F_{MAX}	-
TIM26, TIM27	Kernel	bus_ck	-	-	F_{MAX}	S
	Bus	pclk3	-	-	F_{MAX}	-
USART1	Kernel	pclk2	**0**	USART1SEL	F_{MAX}	A
		ic1_ck	1			
		pll2_q_ck (3)	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		lse_ck	5			
	Bus	pclk2	-	-	F_{MAX}	-
USART3 (4)	Kernel	pclk1	**0**	USART3SEL	F_{MAX}	A
		ic1_ck	1			
		pll2_q_ck	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		lse_ck	5			
	Bus	pclk1	-	-	F_{MAX}	-
USART2, UART4, UART5, UART6	Kernel	pclk1	**0**	USART2SEL UART4SEL UART5SEL UART6SEL	F_{MAX}	A
		ic1_ck	1			
		pll2_q_ck (3)	2			
		hse_ker_ck	3			
		hsi_ker_ck	4			
		lse_ck	5			
	Bus	pclk1	-	-	F_{MAX}	-

Table 119. Peripheral clock distribution summary (continued)

Peripherals	Clock types	Clock sources	Kernel clock MUX		Max. freq. (MHz)(1)	Type (2)
			Pos.	Control bitfield		
UCPD1 (4)	Kernel	hsi_ker_ck	**0**	UCPD1SEL	F_{MAX}	A
		hse_ker_ck	1			
		hsi48_ker_ck	2			
		pll2_q_ck	3			
	Bus	pclk1	-	-	F_{MAX}	-
USB (4)	Kernel	hsi48_ker_ck	**0**	USBSEL	48	A
		hse_ker_ck	1			
		pll1_q_ck	2			
		pll2_q_ck	3			
	Bus	pclk2	-	-	F_{MAX}	-
VREFBUF	Bus	pclk3	-	-	F_{MAX}	-
WWDG	Bus	pclk1	-	-	F_{MAX}	-

1. F_{MAX} value represents the maximum frequency at which the CPU can run. This value depends on the STM32G56x/55x/54x/53x device and can be found on the device datasheet.
2. 'A' means that the kernel clock is asynchronous with respect to bus interface clock.
‘S’ means that the kernel clock is synchronous with respect to bus interface clock.
3. This signal is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
4. These peripherals are not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
5. On STM32G56x/55x/54x/53x devices where the HSI48 is not available, hsi_ker_ck / 2 is provided instead. Refer to the device datasheet for availability.

Clock distribution for UART, USART, and LPUART

The UART and USART kernel clock sources are selected thanks to UARTxSEL[2:0] and USARTxSEL[2:0] in the *RCC kernel clock configuration register (RCC_CCIPR1)*.

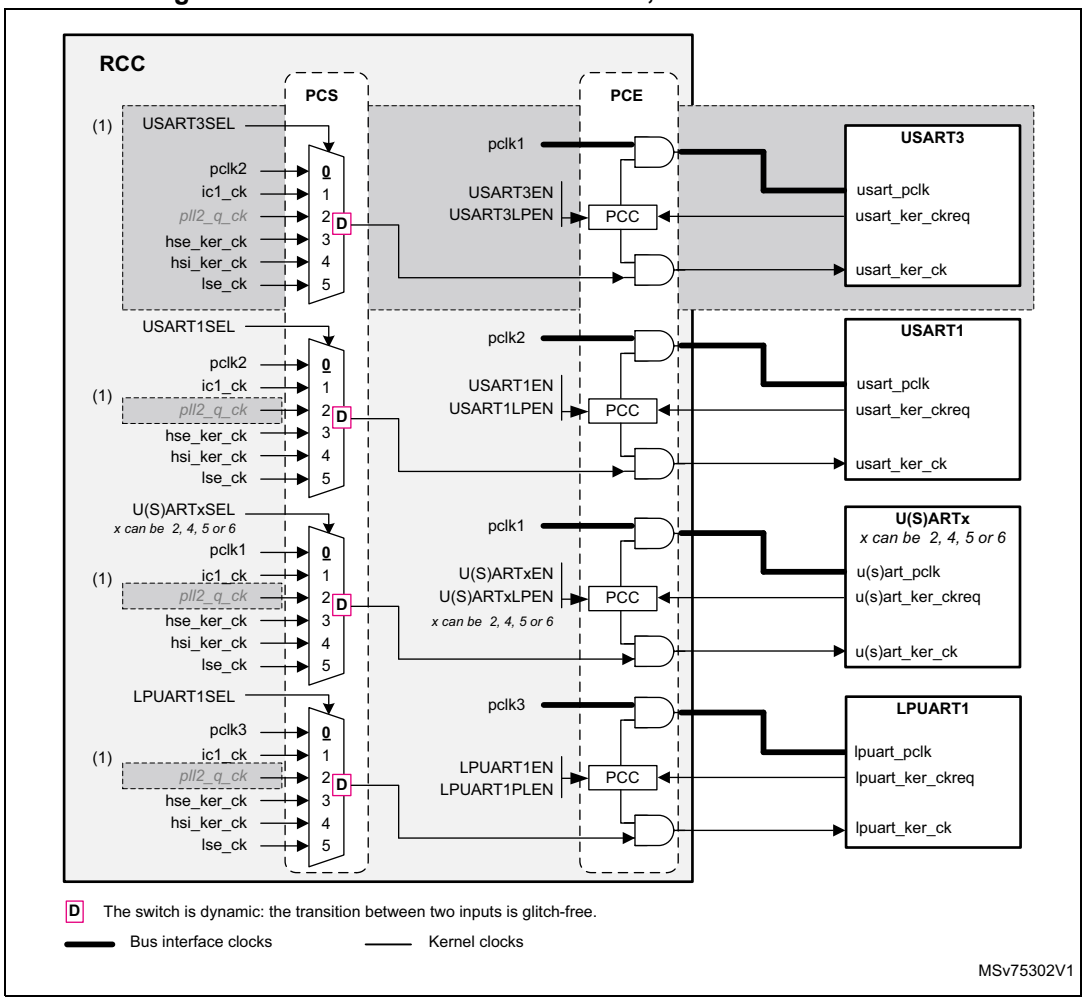
The UARTs, USARTs, and LPUART provide kernel clock request signals handled by the RCC allowing those blocks to get kernel clock even when the device is in Stop mode. Refer to *Table 128: Clock gating rules for generic peripherals* for details on clock gating rules.

This mechanism works if the selected kernel clock source is hsi_ker_ck (automatically turned on, on peripheral request). If lse_ck is selected as kernel clock source, it is always provided to the peripheral if the LSE is enabled.

If UARTs, USARTs or LPUART1 is set to secure via TZSC, the corresponding bitfields xxxSEL, xxxEN, xxxLPEN, and xxxRST can only be changed by secure access. In addition if SPRIV is set, only secure privileged access can modify these bitfields.

If UARTs, USARTs or LPUART1 is set to nonsecure via TZSC, the corresponding bitfields xxxSEL, xxxEN, xxxLPEN, and xxxRST can be changed by secure or nonsecure access. In addition, if NSPRIV is set, only nonsecure or secure privileged access can modify these bitfields.

Figure 60. Clock distribution for UARTs, USARTs and LPUART



1. This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
2. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

All peripheral clocks of the cryptographic subsystem must be disabled before issuing a software reset to any CCB/PKA/RNG/AES/SAES/HASH peripheral of this subsystem.

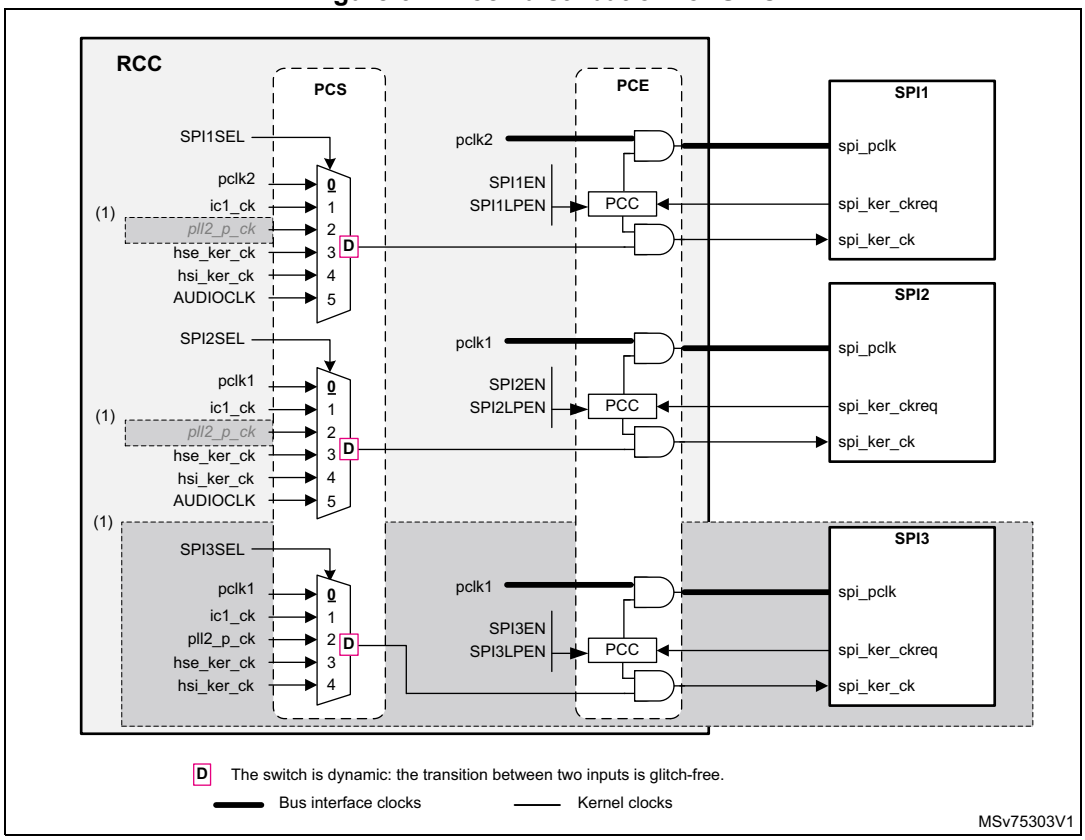
For any CCB/PKA/RNG/AES/SAES/HASH peripheral of this subsystem, the software must ensure that peripheral is not busy (by polling on the BUSY bit flag of the related CCB/PKA/RNG/AES/SAES/HASH peripheral) before disabling the peripheral clock.

Clock distribution for SPIs

The SPI kernel clock sources are selected through SPIxSEL[2:0] in the *RCC kernel clock configuration register (RCC_CCIPR3)*. SPI with I2S feature also supports audio applications. For audio use-cases, the application can select the RC oscillator source hsi_ker_ck for the power-critical applications that do not request an accurate clock, or use hse_ker_ck, PLL outputs or AUDIOCLK when accurate frequency is needed.

The SPIs provide kernel clock request signals allowing these blocks to get kernel clock even when the device is in Stop mode. This mechanism works if the selected kernel clock source is hsi_ker_ck. Refer to *Table 128: Clock gating rules for generic peripherals* for details on clock gating rules.

Figure 61. Clock distribution for SPIs



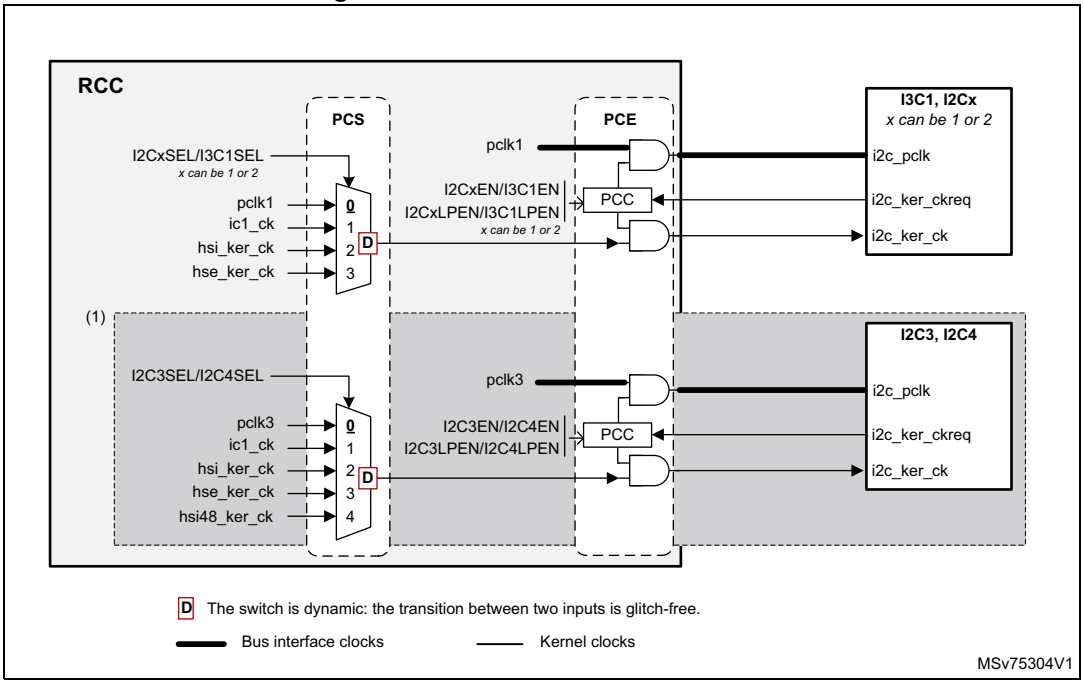
1. This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
2. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

Clock distribution for I2C and I3C

The I2Cs and I3C1 kernel clock sources are selected through I2CxSEL[2:0] and I3C1SEL[2:0], respectively. These bitfields belong to the *RCC kernel clock configuration register (RCC_CCIPR4)*.

The I2Cs and I3C1 provide a kernel clock request signals handled by the RCC allowing those blocks to get kernel clock even when the device is in Stop mode. This mechanism works if the selected kernel clock source is hsi_ker_ck. Refer to *Table 128: Clock gating rules for generic peripherals* for details on clock gating rules.

Figure 62. Clock distribution for I2Cs



1. This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
2. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

Clock distribution for TIMs, HRTIM1 and LPTIM1

The timers (TIMs) use kernel clocks (rcc_tim_ker_ck) that are phase-aligned with the bus interface clock. The TIMs can work at a frequency higher than their bus interface clock, depending on the division factors configured through PPREx. For example if PPRE1 performs a division by 4, then TIM2 kernel clock is set to F_{HCLK}, and the bus interface clock to F_{HCLK} / 4.

The HRTIM1 has a different mechanism allowing it to work at a higher rate than the CPU. As for the TIMs, the kernel clock (rcc_hrtim_ker_ck) must be phase-aligned with the bus interface clock.

The kernel clock frequency depends on:

- The APB2 prescaler (PPRE2), and
- The CPU prescaler HPRE

A logic, named LUT in the drawing below, limits the ratio between the kernel clock frequency and the bus interface clock to ×32.

As shown in *Table 120*, if PPRE2 performs a division by 4 and HPRE a division by 2, then the HRTIM1 kernel clock frequency is eight times faster than the bus interface clock.

Table 120. HRTIM1 clock multiplication ratio (1)

<table>
<tr><th colspan="2" rowspan="2">HRTIM clock multiplication ratio</th><th colspan="5">PPRE2 division ratio</th></tr>
<tr><th>1</th><th>2</th><th>4</th><th>8</th><th>16</th></tr>
<tr><td rowspan="9">HPRE division ratio</td><td>1</td><td>× 1</td><td>× 2</td><td>× 4</td><td>× 8</td><td>× 16</td></tr>
<tr><td>2</td><td>× 2</td><td>× 4</td><td>× 8</td><td>× 16</td><td rowspan="5">× 32</td></tr>
<tr><td>4</td><td>× 4</td><td>× 8</td><td>× 16</td><td rowspan="4">× 32</td></tr>
<tr><td>8</td><td>× 8</td><td>× 16</td><td rowspan="4">× 32</td></tr>
<tr><td>16</td><td>× 16</td><td rowspan="4">× 32</td></tr>
<tr><td>64</td><td rowspan="4">× 32</td></tr>
<tr><td>128</td></tr>
<tr><td>256</td></tr>
<tr><td>512</td></tr>
</table>

1. The multiplication ratio is the ratio between the HRTIM kernel clock and the bus interface clock F_{ck_tim} / F_{clk_apb}.

The LPTIM1 can have a kernel clock fully asynchronous from the bus interface clock. A clock switch controlled by LPTIM1SEL[2:0] allows the application to select a wide variety of kernel clock sources.

TIMs and HRTIM1 cannot work in Stop mode because the system clock is no longer present in this mode (see *Table 121*).

Table 121. Clock gating rules for TIMs and HRTIM1

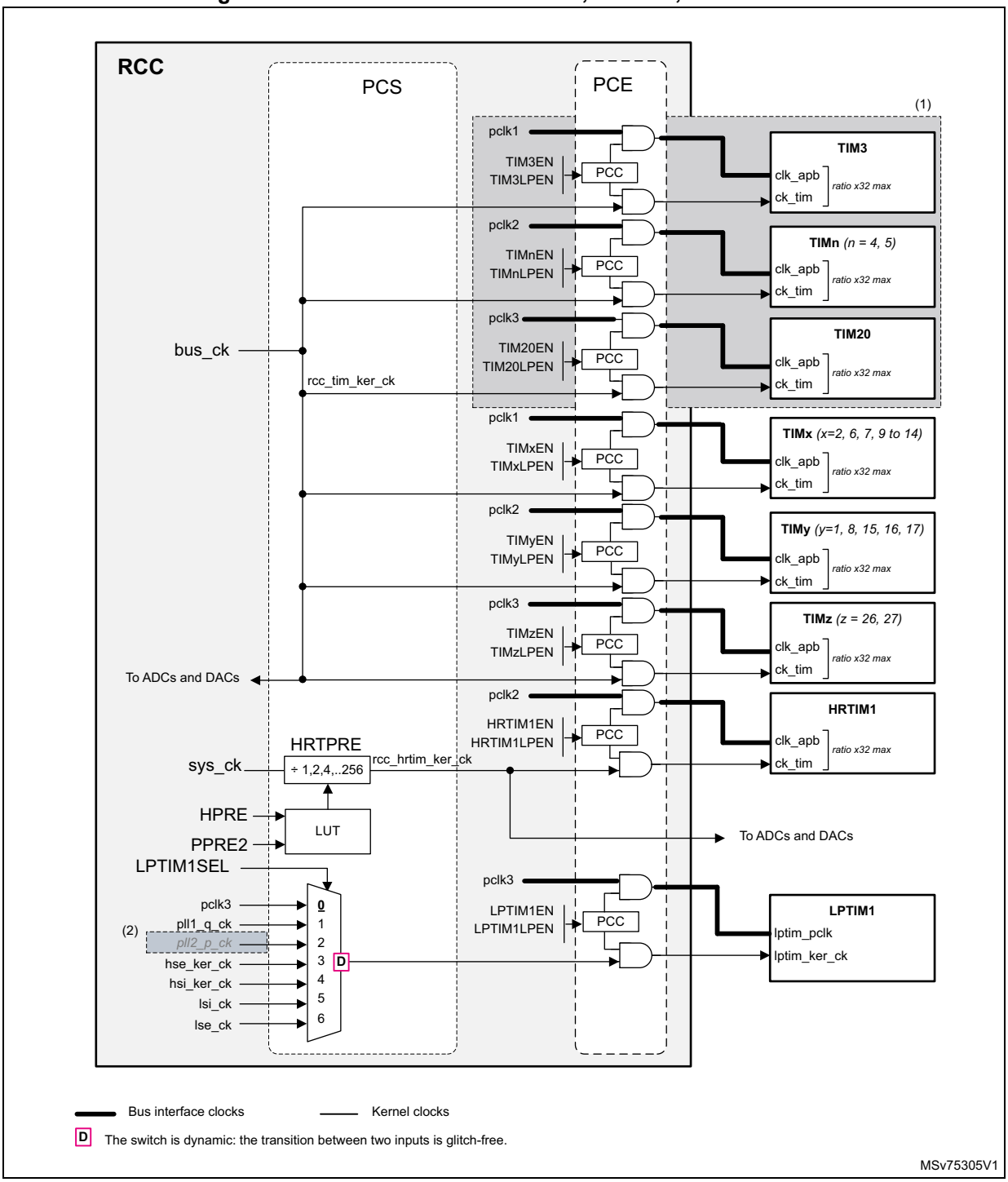
TIMxEN (HRTIMxEN)	TIMxLPEN (HRTIMxLPEN)	CPU Modes	-	Kernel clock (ck_tim)	Bus clock (clk_apb)	Comments
0	X	X	→	-	-	No clock provided
1	X	Run	→	CK	CK	Clock provided
1	0	Sleep	→	-	-	No clock provided
1	1		→	CK	CK	Clocks provided
X	X	Stop	→	-	-	No clock provided

The LPTIM1 can work in Stop mode if the selected kernel clock is either:

- lse_ck, and the LSE is enabled, or
- lsi_ck, and the LSI is enabled, or
- hsi_ker_ck, and the HSI is enabled

Refer to *Section : Generic clock gating rules* for details on clock gating rules.

Figure 63. Clock distribution for TIMs, HRTIM1, and LPTIM1



1. This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
2. This signal is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
3. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

Clock distribution for POSENC1 and POSENC2

These blocks are not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

The POSENC1 and POSENC2 kernel clock sources are selected through POSEN1SEL[2:0] and POSENC2SEL[2:0], respectively. These bitfields belong to the *RCC kernel clock configuration register (RCC_CCIPR5)*.

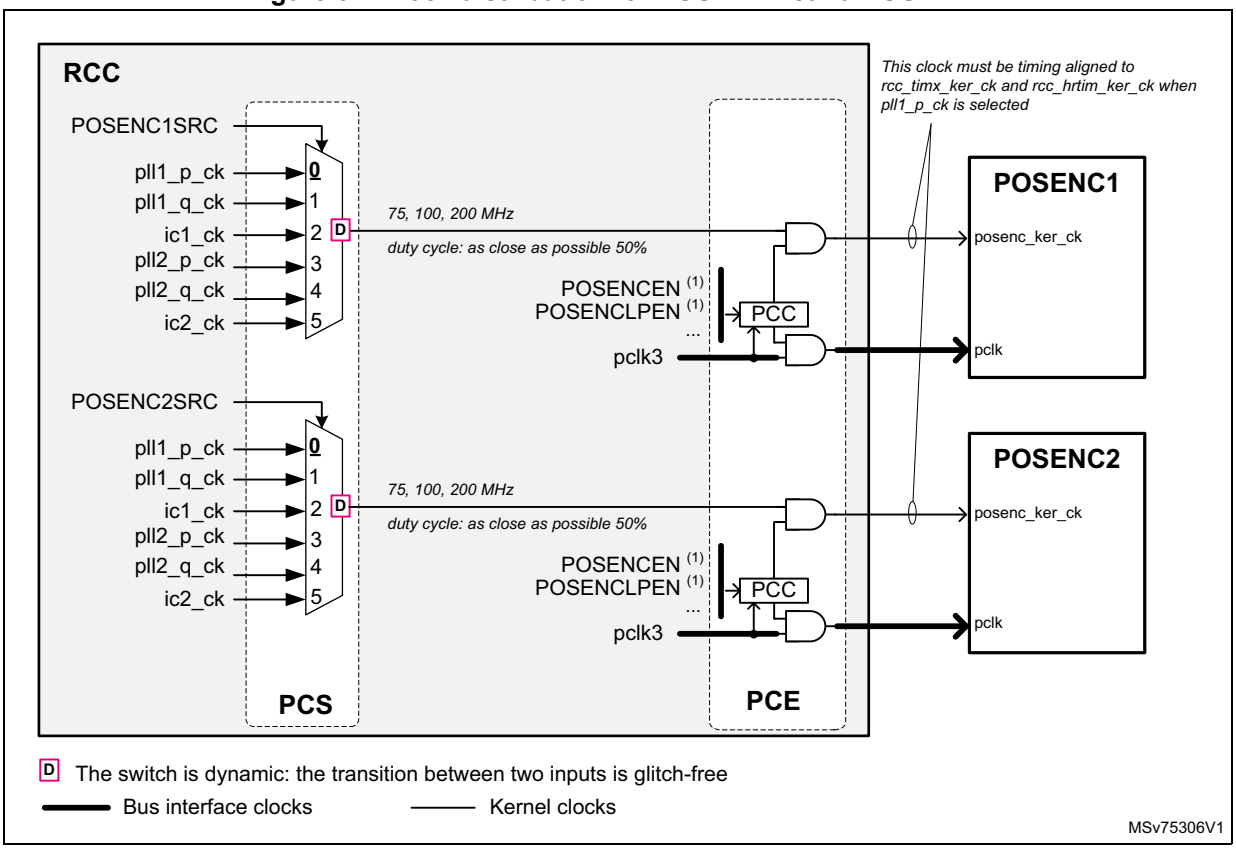
There is no kernel clock request mechanism for POSENC blocks, so they cannot work when the device is in Stop mode. Refer to *Section : Generic clock gating rules* for details.

The POSENC blocks support Endat3 and Hyperface DSL interfaces. Only one interface can be active at a given time. *Table 122* gives the allowed clock frequencies and constraints for each interface.

Table 122. POSENC1 and POSENC2 clock constraints

Protocol	posenc_ker_ck frequencies (MHz)	Duty cycle (%)	Accuracy (ppm)	Other constraints
Endat 3	100 or 200	50	100	APB clock higher than 48 MHz and lower than posenc_ker_ck
Hiperface DSL	75	50	100	

Figure 64. Clock distribution for POSENC1 sand POSENC2



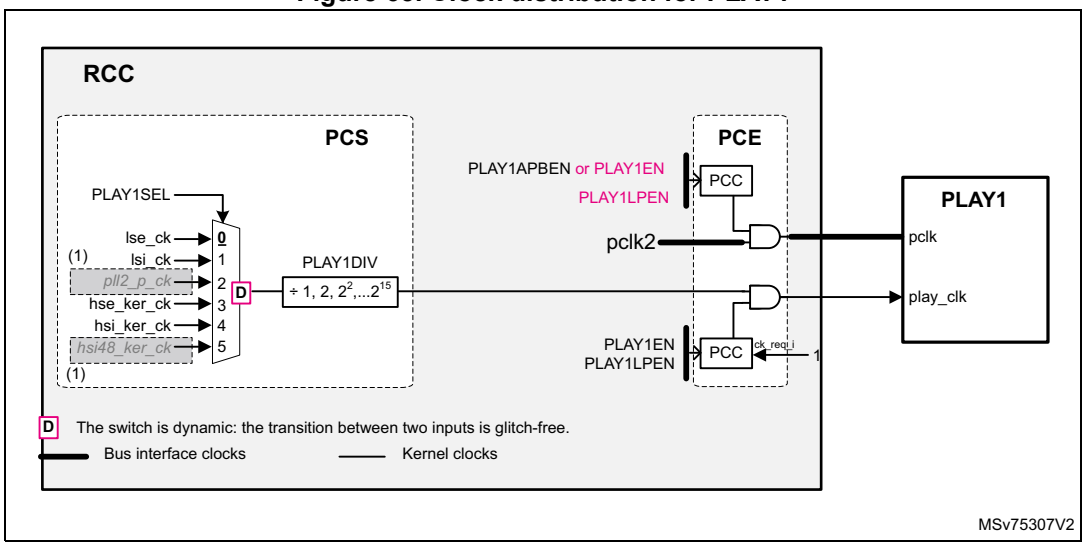
1. PCC stands for peripheral clock control, PCC blocks also receives security and CPU mode signals. They are not shown to simplify the drawing.

Clock distribution for PLAY1

The PLAY1 is a dual-clock domain: it requires a bus clock interface and a kernel clock. The bus clock is required to configure the PLAY1 but also in functional mode.

The PLAY1 kernel clock sources are selected through the PLAY1SEL[2:0]. In addition, a prescaler is available to adjust the kernel clock rate. The prescaler value is adjusted via PLAY1DIV[3:0]. Both PLAY1DIV[3:0] and PLAY1SEL[2:0] are located in the *RCC kernel clock configuration register (RCC_CCIPR3)*.

Figure 65. Clock distribution for PLAY1



1. This signal is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
2. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

In Run mode, the bus clock (pclk2) is provided to the PLAY1 if the PLAY1APBEN or the PLAY1EN bit is set. In addition, the bus clock is also provided in Sleep mode if PLAY1LPEN is set. In Stop mode, this clock is not provided.

In Run mode, the kernel clock is provided to the PLAY1 if PLAY1EN is set. In addition, the kernel clock is also provided in Sleep mode if PLAY1LPEN is set. In Stop mode, this clock is also provided if the selected kernel clock is hsi_ker_ck, lse_ck, or lsi_ck.

If the hsi_ker_ck clock is selected, the RCC automatically switches the HSI oscillator ON (see *Table 123* and *Table 124*).

Table 123. Clock gating rules for PLAY1 bus clock

PLAY1APBEN or PLAY1EN	PLAY1LPEN	CPU Modes	-	PLAY1 bus clock	Comments
0	X	X	→	-	No clock provided
1	X	Run	→	CK	Clocks provided

Table 123. Clock gating rules for PLAY1 bus clock (continued)

PLAY1APBEN or PLAY1EN	PLAY1LPEN	CPU Modes	-	PLAY1 bus clock	Comments
1	0	Sleep	→	-	No clock provided
1	1		→	CK	Clocks provided
X	X	Stop	→	-	No clock provided

Table 124. Clock gating rules for PLAY1 kernel clock

PLAY1EN	PLAY1LPEN	PLAY1SEL	CPU Modes	-	PLAY1 kernel clock	Comments
0	X	X	X	→	-	No kernel clock provided
1	X	X	Run	→	CK	Kernel clocks provided
1	0	X	Sleep	→	-	No kernel clock provided
1	1	X		→	CK	Kernel clocks provided
1	0	X	Stop	→	-	No kernel clock provided
1	1	Other clock selected		→	-	No kernel clock provided
1	1	lsi_ck or lse_ck or hsi_ker_ck		→	CK	Kernel clocks provided

Clock distribution for ADC and DAC

The kernel clock source of all ADCs is selected through ADCSEL[2:0], and can be divided directly in the RCC by the ADCDIV[3:0]. ADCSEL[2:0] and ADCDIV[3:0] are located in the *RCC kernel clock configuration register (RCC_CCIPR5)*.

For each DAC, the kernel clock source can be selected through DACxSEL[2:0] in the *RCC kernel clock configuration register (RCC_CCIPR2)*. Some DACs also offer an additional low-power clock (LSI or LSE) to operate in sample and hold mode, available in Stop mode. This clock is selected via the DACxSHSEL bits of the *RCC kernel clock configuration register (RCC_CCIPR2)*.

There is no kernel clock request mechanism for DAC and ADC blocks. The ADCs cannot work when the device is in Stop mode. Similarly, most of the DACs cannot work when the device is in Stop mode except the hold function clocked by LSE or LSI for DACs receiving these clocks.

Refer to *Section : Generic clock gating rules* for details on clock gating rules.

To optimize analog-to-digital conversion, it is strongly recommended to adjust the kernel clock frequency of the ADCs by adjusting ADCDIV, instead of changing the default values of the prescalers located into each ADC block. It guarantees that all ADCs sample data without disturbing each other. The divider ADCDIV provides a 50% duty cycle clock signal. In order to change the division value the application must perform the following sequence:

1. Disable all ADCs by clearing ADC12EN and ADC3EN.
2. Change the ADCDIV[3:0] division factor.
3. Enable the ADCs by setting ADC12EN and/or ADC3EN.

If the application requires that the ADCs are precisely triggered by a TIMx timer without any uncertainty (fixed trigger latency); ADCSEL must select rcc_tim_ker_ck as clock source (position 0).

Note: To solve trigger uncertainty, the ADCDIV[3:0] divider delivers a synchronization signal to the timers.

If the application requires that the DACs are precisely triggered by a TIMx timer without any uncertainty (fixed trigger latency), DACxSEL and HPRE must be programmed as follows:

- Clear DACxSEL (HPRE value is don’t care), or
- Set both DACxSEL and HPRE.

In all other configurations, there might be a small trigger uncertainty due to clock resynchronization.

If DACxSEL is set, and HPRE has to be changed, it is strongly recommended to first disable the corresponding DAC (via DACxEN bits), then change the HPRE value, and enable again the DAC.

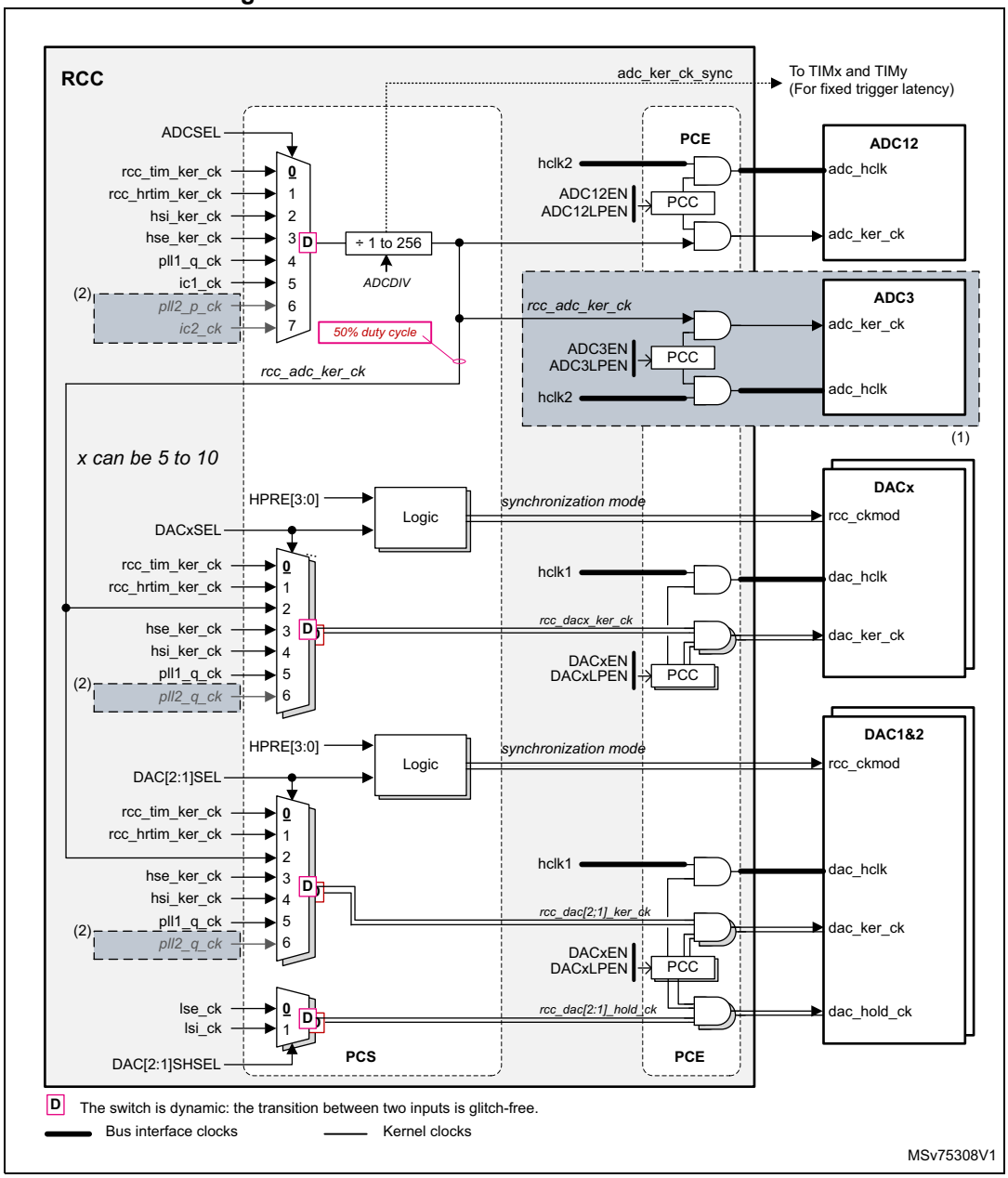
If the kernel clock source of the DACx has to be changed, the following sequence must be respected by the application:

1. Disable the DACx clocks by clearing DACxEN.
2. Program DACxSEL to the desired value.
3. Enable the DACx clocks by setting DACxEN.

Note: If ADCs or DACs are triggered by timers, and the ADC or DAC clock sources are asynchronous with respect to timer clocks, there is an uncertainty of the trigger instant due to the resynchronization between the two clock domains.

If ADC12 or ADC3 is set to secure via TZSC, the common bitfields ADCSEL[2:0] and ADCDIV[3:0] can only be changed by secure accesses. In addition if SPRIV is set, only secure privileged access can modify these bitfields.

Figure 66. Clock distribution for ADCs and DACs



1. This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
2. This signal is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
3. PCC stands for peripheral clock control, PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

Clock distribution for cryptographic blocks

The configuration of cryptographic blocks depends on the STM32G56x/55x/54x/53x device. Some devices feature a CCB subsystem including RNG, PKA, SAES, AES, HASH, others offer only a subset including RNG, PKA, AES, and HASH, while some others do not have cryptographic blocks (refer to the device datasheet for availability).

Configuration with CCB subsystem (STM32G55x/56x devices)

The CCB block shares the same clock as the AHB bus clock. The CCB automatically requests the RCC to enable clocks of the RNG, SAES, and PKA, using dedicated clock request signals.

The PKA receives a clock when the tamp_erase_req or rdp_erase_ram becomes active, whatever the values of PKAEN and PKALPEN (see *Table 125* for details).

Table 125. PKA clock gating rules

PKAEN	PKALPEN	tamp_erase_req	rdp_erase_ram	CPU state	-	pka_hclk
0	X	0	0	CRun or CSleep	→	-
X	X	1	X			CK
X	X	X	1			CK
1	X	0	0	CRun		CK
	0			CSleep		-
	1					CK
X	X	X	X	CStop		-

The RNG block uses the hsi48_ker_ck as kernel clock. When the CCB activates the rng_hclk_en, the RCC provides to the RNG the kernel clock and bus clock even if the RNGEN bit is cleared. If the HSI48 oscillator is disabled, the RCC automatically enables it, waits until the clock is ready, and provides it to the RNG.

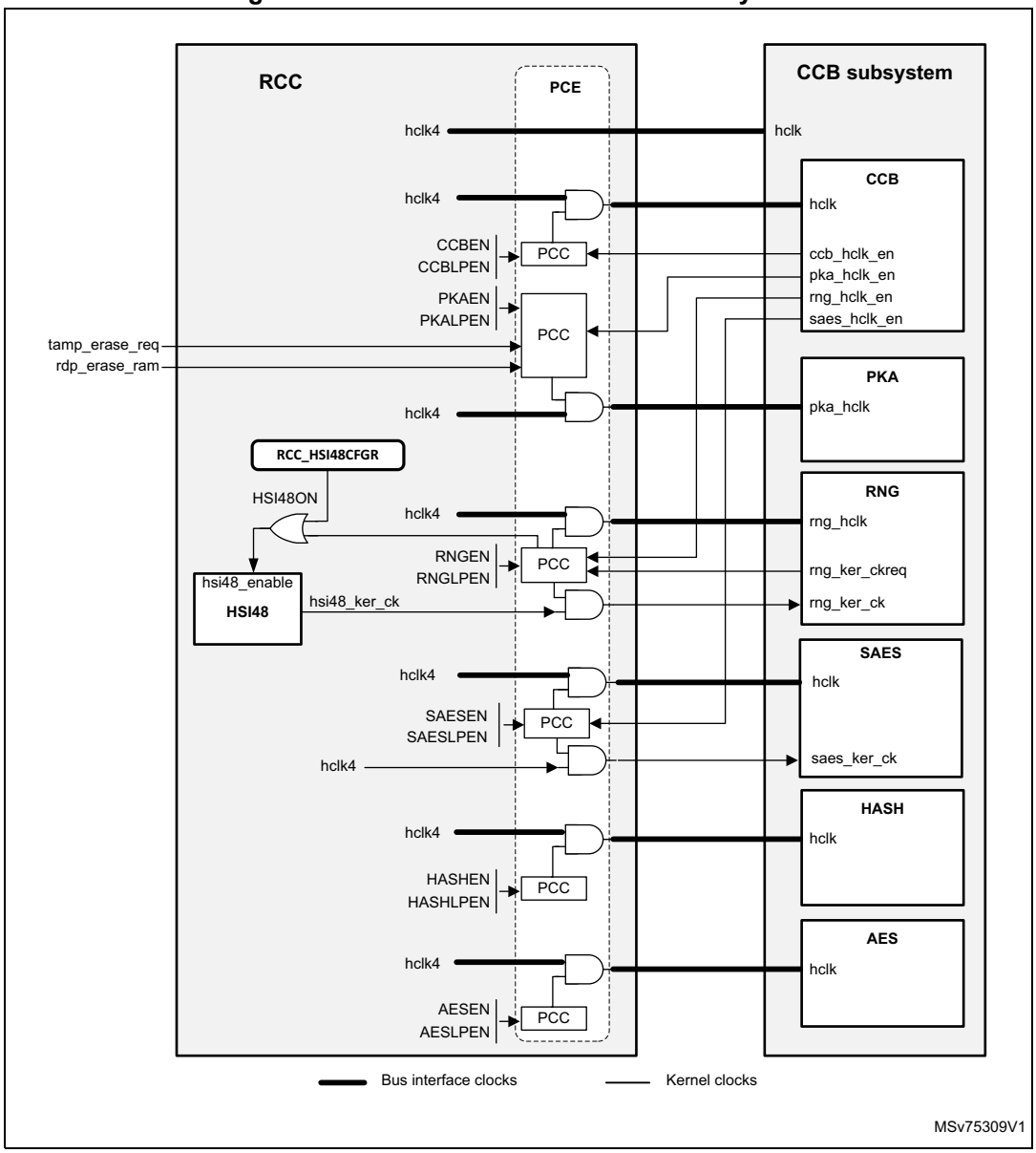
The logic that controls the clock of the RNG is different from standard peripherals. When enabled, the RNG has to assert its kernel clock request to get a kernel clock. For example, if the SAES needs the RNG resource, the RNG must also be enabled (see *Table 126*).

Table 126. RNG clock gating rules with CCB

RNGEN	RNGLPEN	rng_ker_ckreq	rng_ck_en	CPU state	-	rng_ker_ck	rng_hclk
0	X	X	0	CRun or CSleep	→	-	-
X	X	X	1			CK	CK
1	X	0	0	CRun		-	CK
1	X	1	0			CK	CK
1	0	X	0	CSleep		-	-
1	1	0	0			-	CK
1	1	1	0			CK	CK
1	X	X	X	CStop		-	-

Note: The RNG kernel clock is the hsi48_ker_ck.

Figure 67. Clock distribution for CCB subsystem



1. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.
2. All peripheral clocks of the cryptographic subsystem must be disabled before issuing a software reset to any CCB/PKA/RNG/AES/SAES/HASH peripheral of this subsystem. For any CCB/PKA/RNG/AES/SAES/HASH peripheral of this subsystem, the software must ensure that the peripheral is not busy (by polling on the BUSY bit flag of the related CCB/PKA/RNG/AES/SAES/HASH peripheral) before disabling the peripheral clock.

Configuration with AES, RNG, PKA, and HASH (STM32G53x/54x devices)

The PKA receives a clock when the tamp_erase_req or rdp_erase_ram becomes active, whatever the values of PKAEN and PKALPEN (see *Table 125: PKA clock gating rules* for details).

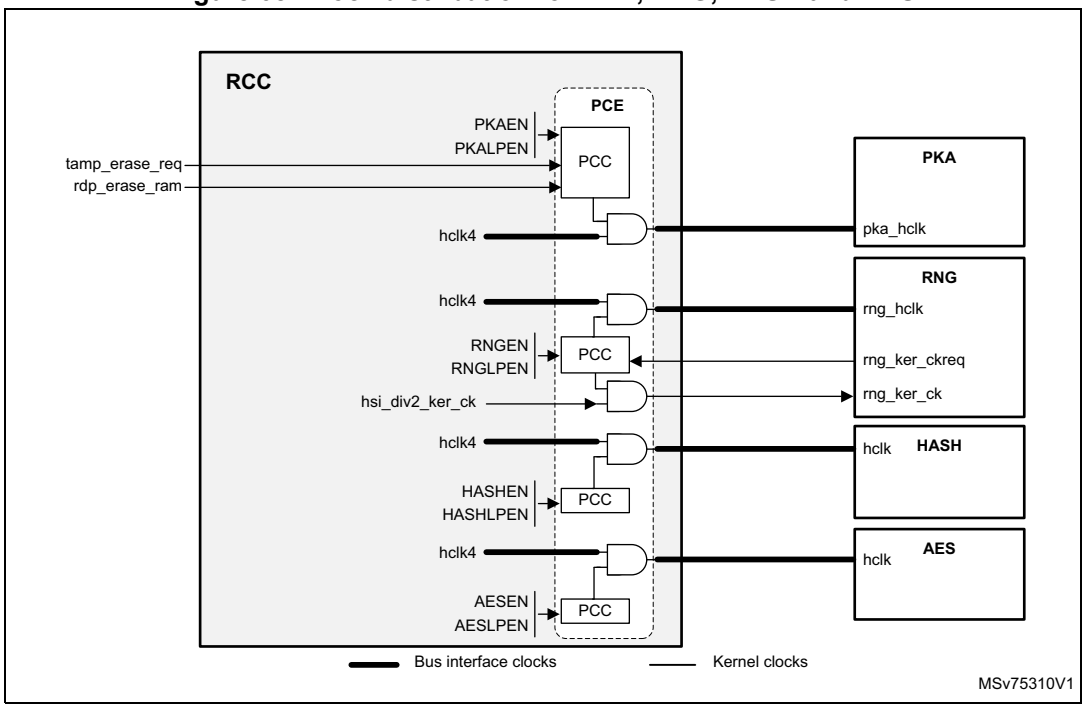
The RNG block uses the hsi_div2_ker_ck as kernel clock. The logic controlling the clock of the RNG is different from standard peripherals. When enabled, the RNG has to assert its kernel clock request to get a kernel clock. If the HSI oscillator is disabled, the RCC automatically enables it, waits until the clock is ready, and provides it to the RNG (see *Table 127*).

Table 127. RNG clock gating rules without CCB

RNGEN	RNGLPEN	rng_ker_ckreq	CPU state	-	rng_ker_ck	rng_hclk
0	X	X	CRun or CSleep	→	-	-
1	X	0	CRun		-	CK
1	X	1			CK	CK
1	0	X	CSleep		-	-
1	1	0			-	CK
1	1	1			CK	CK
1	X	X	CStop		-	-

Note: The RNG kernel clock is the hsi_div2_ker_ck.

Figure 68. Clock distribution for PKA, RNG, HASH and AES



1. PCC stands for peripheral clock control. PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

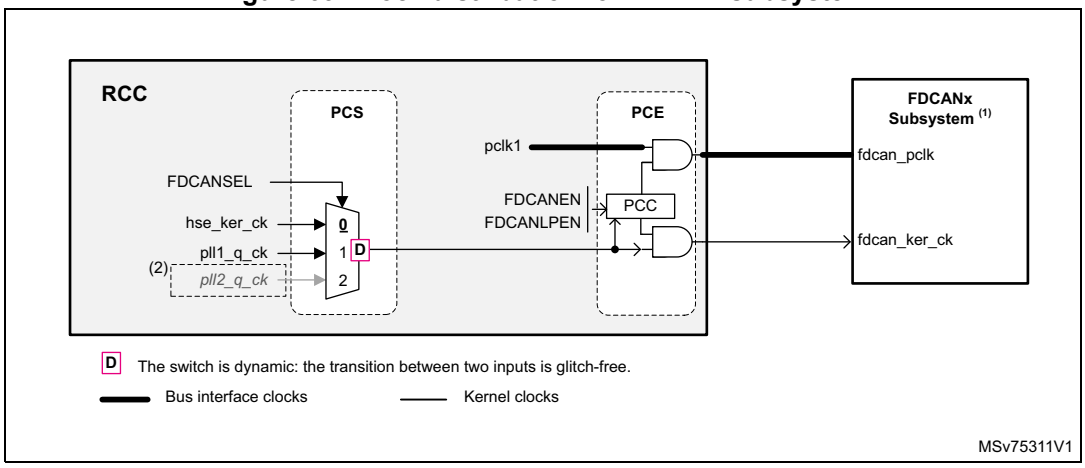
Clock distribution for FDCAN subsystem

STM32G56x/55x/54x/53x FDCAN subsystem embeds one or three FDCAN instances, and all sharing the same bus clock and the same kernel clock. The FDCAN kernel clock is selected by the FDCANSEL[1:0] bitfield located into *RCC kernel clock configuration register (RCC_CCIPR5)*.

There is no kernel clock request mechanism for FDCAN blocks, so they cannot work when the device is in Stop mode. Refer to *Table 128: Clock gating rules for generic peripherals* for details on clock gating rules.

If one of the FDCAN instances (FDCAN1, FDCAN2 or FDCAN3) is configured to be secure via TZSC, the FDCANSEL[1:0], FDCANRST, FDCANEN, and FDCANLPEN bitfields can only be changed by secure access. In addition, if SPRIV is set, only secure privileged access can modify these bitfields.

Figure 69. Clock distribution for FDCAN subsystem



1. The STM32G53x/54x devices feature only one FDCAN instance, while others have three instances. Refer to the device datasheet for availability.
2. This signal is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
3. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

Note: The frequency of the FDCAN bus clock (fdcan_pclk) must be higher than the kernel clock frequency (fdcan_ker_ck).

Clock distribution for XSPI1

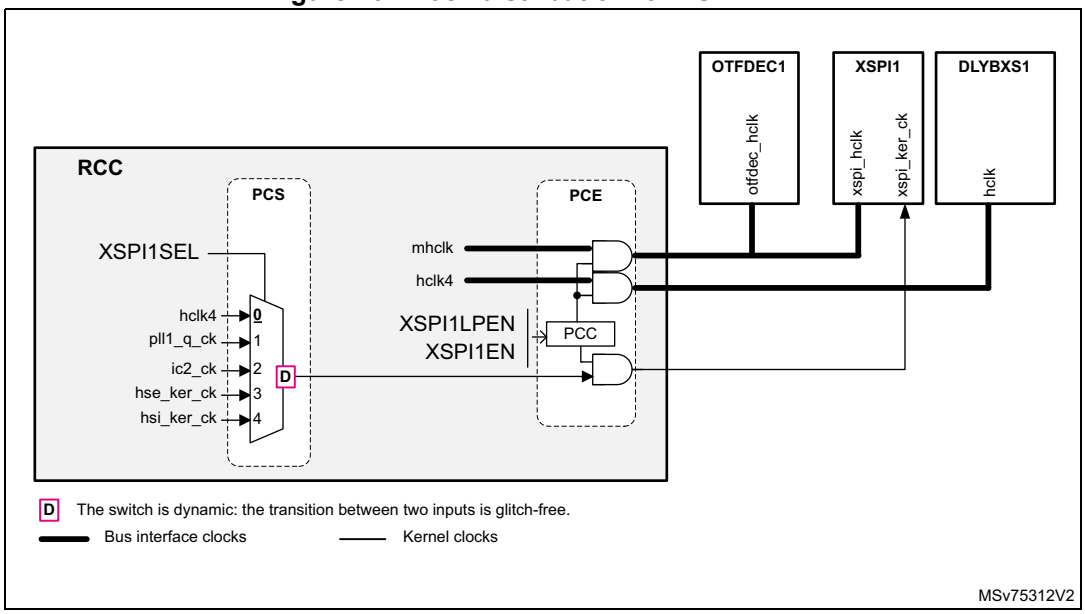
This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

The XSPI1 kernel clock sources are selected through XSPI1SEL[2:0] in the *RCC kernel clock configuration register (RCC_CCIPR4)*.

There is no kernel clock request mechanism for XSPI1, so they cannot work when the device is in Stop mode. Refer to *Table 128: Clock gating rules for generic peripherals* for details on clock gating rules.

The kernel clock must be lower or equal to the bus clock.

Figure 70. Clock distribution for XSPI1 (1) (2)



1. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.
2. This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

Clock distribution for USB and UCPD1

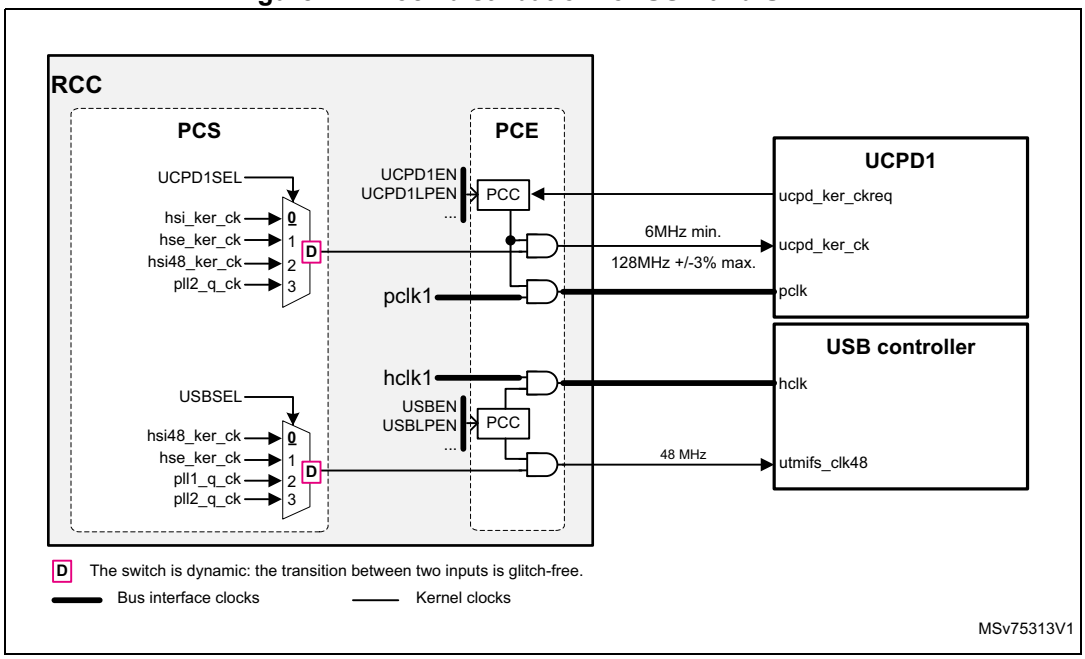
This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

USB and UCPD1 kernel clocks are selected through USBSEL[1:0] and UCPD1SEL[1:0], respectively. These bitfields are located in the *RCC kernel clock configuration register (RCC_CCIPR4)*.

The UCPD1 kernel clock must not be lower than 6 MHz, and must not exceed 100 MHz, with a precision of ± 3%.

The UCPD1 provides a kernel clock request signal allowing this block to get kernel clock even when the device is in Stop mode. This mechanism works if the selected kernel clock source is hsi_ker_ck. Refer to *Table 128: Clock gating rules for generic peripherals* for details on clock gating rules.

Figure 71. Clock distribution for USB and UCPD1



1. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.
2. This block is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

Clock distribution for RTC/TAMP

The kernel clock source, used by the RTC and TAMP blocks, can be one of the following:

- The hse_rtc_ck (hse_ker_ck divided by a programmable prescaler)
- The lse_ck
- The lsi_ck clock

The source clock is selected by programming the RTCSEL[1:0] bits of the *RCC RTC domain control register (RCC_RTCCR)* and the RTCPRE[5:0] bits of the *RCC clock configuration register1 (RCC_CFGR1)*.

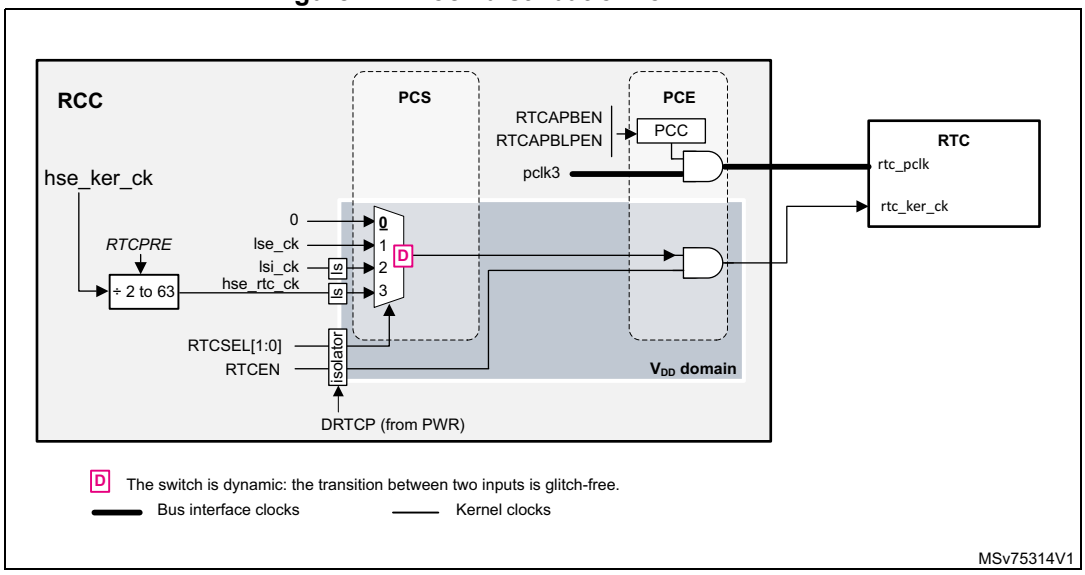
Note that the content of RTCEN and RTCSEL[1:0] is propagated to the V_{DD} domain only when DRTCP is set.

RTCSEL[1:0] can be written only once after an RTC domain reset, or after a failure detection on LSE. RTCSEL[1:0] must not be modified if LSECSSON is cleared.

The system must always be configured so as to provide a bus clock frequency (pclk3) greater than or equal to the clock frequency provided to rtc_ker_ck.

The TAMP does not require any kernel clock if only the backup registers are used, with tampers in edge detection mode. All other tamper detection modes require a kernel clock (refer to *Section 50: Tamper and backup registers (TAMP)*).

Figure 72. Clock distribution for RTC/TAMP



1. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

Note: The hse_rtc_ck clock is also connected to timers for applications requesting a check of HSE frequency before using it as system clock. See Section 11.5.15: Clock frequency measurement using TIMx.

The LSE clock belongs to the RTC domain.

- If LSI or LSE is selected as the RTC and TAMP clock, the RTC state is not guaranteed if the V_{DD} supply is powered off:
  - The TAMP block can remain functional if used in a mode that does not need any kernel clock.
  - The RTC works in Stop and Standby modes.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_{DD} or V_{CORE} supplies are powered off:
  - The TAMP block can remain functional if used in a mode that does not need any kernel clock.
  - The RTC does not work in Stop and Standby modes.

When the RTC and TAMP clock is LSE or LSI, the RTC remains clocked and functional under system reset.

The RTC bus interface clock (APB clock) is enabled through the RTCAPBEN and RTCAPBLPEN bits of the *RCC APB3 peripheral clock enable register (RCC_APB3ENR)* and *RCC APB3 peripheral low-power clock enable register (RCC_APB3LPENR)*, respectively.

Note: To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency (FAPB < 7 × FRTCLCK), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC_TR gives the same result than the first one. Otherwise, a third read access must be performed.

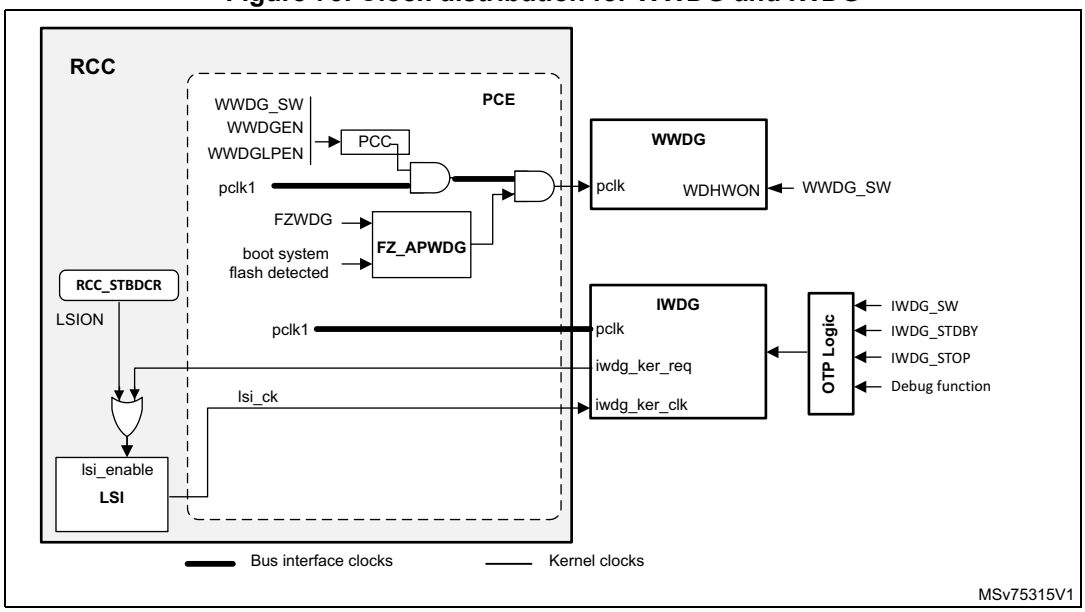
Clock distribution for WWDG and IWDG

The RCC provides the clock for all the watchdog blocks available in the device. The independent watchdog IWDG uses the clock generated by LSI as kernel clock, while the window watchdog WWDG uses the APB clock as kernel clock.

The IWDG can be started by using the IWDG_SW option byte or by software. When started, the LSI is forced ON and cannot be disabled. After the LSI oscillator startup delay, the clock is provided to the IWDG.

The WWDG can be started by using the WWDG_SW option byte or by software. When started by option byte, the APB clock is forced ON, as well as the WWDGEN and WWDGLPEN. Whatever the way the WWDG is started, the software cannot stop it by clearing the WWDGEN bit.

Figure 73. Clock distribution for WWDG and IWDG



1. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

The WWDG is frozen in the following cases:

- When the MCU goes to Stop mode.
- If the WWDG is not started by option bytes, clearing the WWDGLPEN bit freezes the WWDG when the MCU goes to Sleep.
- After a system reset, if the product reboots from system flash memory, the WWDG is frozen during the execution of the system flash code.

The FZ_APWDG logic blocks are used to freeze and unfreeze the watchdog clock according to the boot sequence.

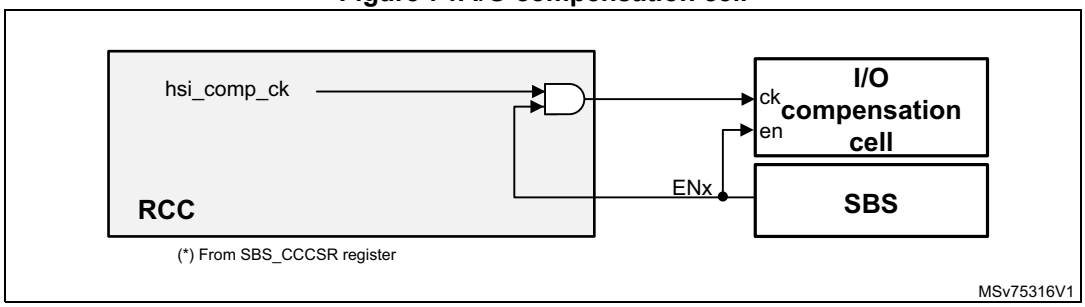
Clock distribution for the I/O compensation cell

The I/O compensation cell is used to calibrate the I/O slew rate. For that purpose, the RCC provides the hsi_comp_ck clock to the I/O compensation cell block.

When the compensation mechanism needs to be activated, the clock must be provided to the I/O compensation cell block, meaning that the application has to set the HSION bit.

When HSION is set, the hsi_comp_ck remains gated until HSIRDY is set, to avoid propagating spurs.

Figure 74. I/O compensation cell



1. PCC stands for peripheral clock control (see *Section 11.5.16: Peripheral clock gating* for details). PCC blocks also receive security and CPU mode signals. They are not shown to simplify the drawing.

The hsi_comp_ck clock is not provided to the I/O compensation cell if the ENx of the SBS_CCCSR register is cleared.

The hsi_comp_ck clock is automatically gated when the system goes to Stop, provided the HSIKERON bit is cleared. Setting the HSIKERON bit allows to maintain the hsi_comp_ck clock enabled even in system Stop.

In system Standby mode, the hsi_comp_ck clock is disabled.

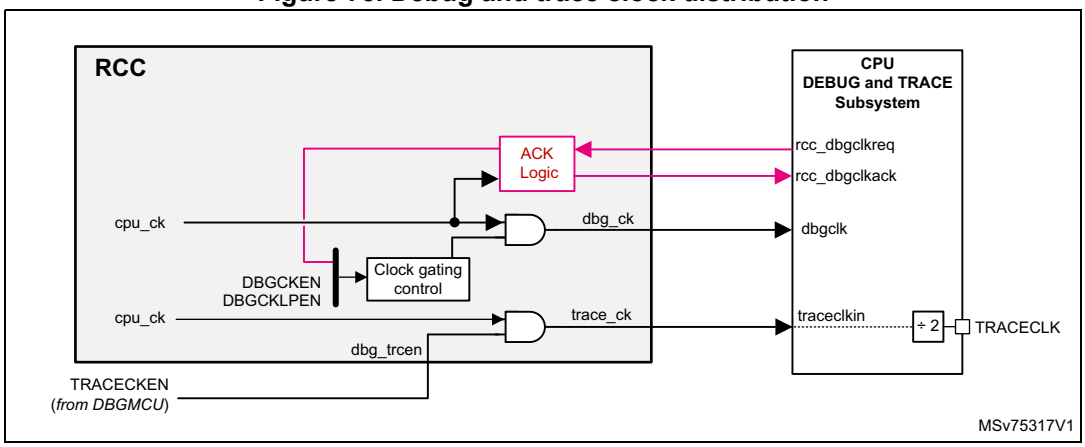
Clock distribution for debug and trace

The clock generation for the debug components is controlled either by the RCC or by the CPU debug and trace subsystem. The clock generation for the trace is controlled via the DBGMCU block.

- The DBGCKEN bit allows the application to provide a clock to the debug components, when the CPU is in Run mode.
- The DBGCKLPEN bit allows the application to provide a clock to the debug components, when the CPU is in Sleep or Stop mode.
- It is also possible to enable the dbg_ck clock via the debug access port using a request/acknowledge handshake.

DBGCKEN and DBGCKLPEN are located in the *RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)* and *RCC AHB3 peripheral low-power clock enable register (RCC_AHB3LPENR)*.

Figure 75. Debug and trace clock distribution



11.5.15 Clock frequency measurement using TIMx

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx:

- Calibrating the HSI with the HSE:

  The RCC provides a dedicated signal hsi_cal_ck to perform the calibration of the HSI oscillator. The hsi_cal_ck represents the HSI oscillator output divided by 1024.

  The signal hsi_cal_ck is connected to at least one timer input capture. Check which timer is connected to hsi_cal_ck in your product.

  For this calibration, it is expected to have the HSE used as the interconnect clock source via PLL1. The number of ck_tim clock periods between consecutive edges of the hsi_cal_ck signal provides a measurement of the HSI clock period. tim_ck is the timer kernel clock.

  The calibration accuracy depends on the frequency ratio between the timer kernel clock (tim_ck) and the hsi_cal_ck clock. The greater the ratio, the more accurate the measurement. The frequency of the hsi_cal_ck can range between ~7 kHz and ~63 kHz. For a timer kernel clock of 50 MHz, the HSI calibration accuracy is about 0.13% (worst case).

  The HSI oscillator has dedicated user-accessible calibration bits to tune the oscillator frequency (see *RCC HSI calibration register (RCC_HSICFGR)*). When the HSI is used via PLLs, the PLL output frequency can also be fine-tuned by adjusting the fractional divider instead of changing HSITRIM[6:0].

- Calibrating the HSI with the LSE:

  The primary purpose of having the LSE connected to a TIMx input capture is to be able to accurately measure the HSI. This requires to use the HSI as system clock source either directly or via PLL1. The number of system clock counts between consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the precision of LSE crystals, we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

  The basic concept consists in providing a relative measurement (HSI/LSE ratio). The precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio, the more accurate the measurement.

  For this purpose, the HSI oscillator has dedicated user-accessible calibration bits (see *RCC HSI calibration register (RCC_HSICFGR)*). When the HSI is used via the PLLx, the system clock can also be fine-tuned by using the fractional divider of the PLLs.

- Calibrating the LSI with the HSI:

  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultralow-power LSI oscillator has a large manufacturing process deviation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, a more accurate RTC timebase timeout (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy can be obtained.

Crystal frequency estimation

The hse_rtc_ck clock (HSE divided by a programmable prescaler) is also connected to a timer to have a rough indication of the external crystal frequency. This can be useful, for instance, to ensure compliance with the IEC 60730/IEC 61335 standards, which require to be able to determine harmonic or subharmonic frequencies (-50/+100% deviations).

To use this feature after a power-on reset, the application must do the following:

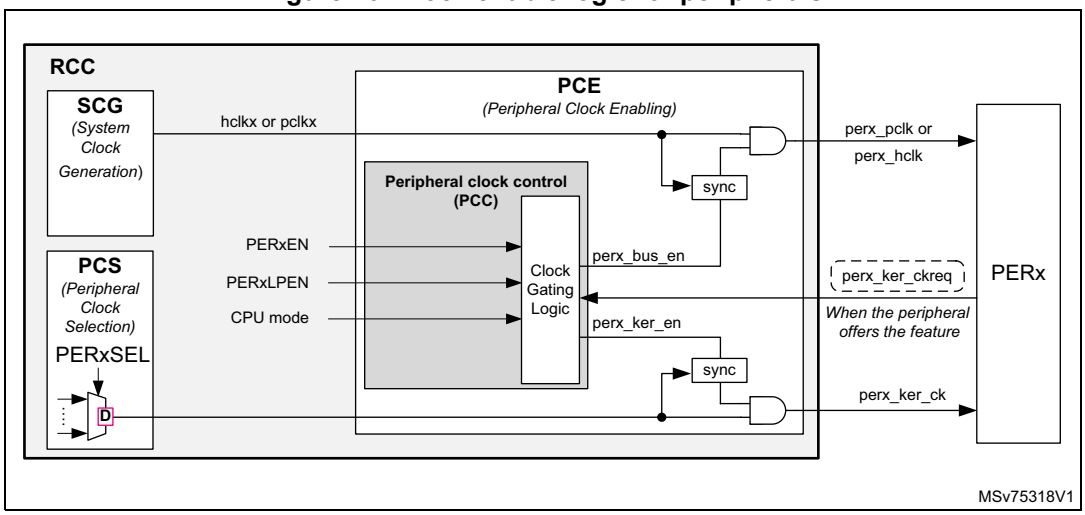
1. Keep HSI as system clock.
2. Program the RTCPRE[5:0] bitfield of the *RCC clock configuration register1 (RCC_CFGR1)* to 0x3F to select the 63 division factor.
3. Program the timer receiving the hse_rtc_ck on one of its input channels, to count the number of HSI periods within a period of hse_rtc_ck.
4. Check that the hse_rtc_ck period value is roughly the one expected.
5. The HSE can then be used as system clock.

11.5.16 Peripheral clock gating

As shown in *Figure 76*, enabling the kernel and bus interface clocks of each peripheral depends on several input signals:

- PERxEN and PERxLPEN bits.
- The CPU mode.
- The kernel clock request (perx_ker_ckreq) of the peripheral itself, when this feature is available.

Figure 76. Clock enable logic for peripherals



Generic clock gating rules

Table 128: Clock gating rules for generic peripherals shows the clock gating rules applying to most of the peripherals:

- UARTs, USARTs, LPUART1
- SPI/I2Ss
- I2Cs, I3Cs
- POSENCs (when available in the product)
- ADCs, DACs
- FDCAN
- XSPI1, USB, UCPD1 (when available in the product)

Table 128. Clock gating rules for generic peripherals

PERxEN	PERxLPEN	PERxSEL	perx_ker_ckreq [1]	CPU Modes	-	perx_ker_ck	perx_pclk or perx_hclk	Comments
0	X	X	X	X	→	-	-	No clock provided to the peripheral, because PERxEN = 0
1	X	X	X	Run	→	CK	CK	Clocks provided: the System is in Run, and PERxEN = 1
1	0	X	X	Sleep	→	-	-	No clock provided: the System is in Sleep, and PERxLPEN = 0
1	1	X	X		→	CK	CK	Clocks provided: the System is in Sleep and PERxEN = PERxLPEN = 1
1	0	X	X	Stop	→	-	-	No clock provided: the PERxLPEN bit is cleared.
1	1	no RC oscillator selected	X		→	-	-	No clock provided: the System is in Stop and **lse_ck** or RC oscillators are not selected.
1	1	lsi_ck or lse_ck	X		→	CK	-	Kernel clock provided: PERxEN = PERxLPEN = 1 and **lsi_ck** or **lse_ck** are selected. Bus interface clock not provided: the System is in Stop. It is up to the application to enable the LSE or LSI if requested.
1	1	HSI oscillator selected	1		→	CK	-	Kernel clock provided: clock request = 1 and PERxEN = PERxLPEN = 1 and an RC oscillator is selected. If the oscillators are OFF, the RCC enable them when a kernel clock request occurs. Bus interface clock not provided: the System is in Stop.
1	1		0		→	-	-	No clock provided: the System is in Stop, and no kernel clock request pending

1. This signal is not provided by all peripherals. For peripherals not handling the kernel clock request, it must be considered set to 0 unless otherwise specified.

As a summary, the kernel clock is provided to the peripherals when the following conditions are met:

1. The CPU is in Run mode, and the peripheral is enabled.
2. The CPU is in Sleep mode and the peripheral is enabled with PERxLPEN = 1.
3. The CPU is in Stop mode. The peripheral is enabled with PERxLPEN = 1. The peripheral generates a kernel clock request, and the selected clock is hsi_ker_ck.
4. The CPU is in Stop mode. The peripheral is enabled with PERxLPEN = 1 and the kernel source clock of the peripheral is lse_ck or lsi_ck.

The bus interface clock is provided to the peripherals only when conditions 1 or 2 are met.

Clock gating rules for other peripherals

- For TIMs and HRTIM1, see *Section : Clock distribution for TIMs, HRTIM1 and LPTIM1*.
- For PLAY1, see *Section : Clock distribution for PLAY1*.
- For cryptographic blocks, see *Section : Clock distribution for cryptographic blocks*.

11.5.17 Bus and CPU clocks

AHB and APB peripheral clocks, including the DMA clock, can be disabled by software via the xxxDIS bits located in the *RCC CPU domain clock configuration register 2 (RCC_CFGR2)*.

In Sleep mode, the CPU clock is stopped. The memory interface clocks (flash memory, caches, and all SRAM interfaces) can be stopped by software during Sleep mode.

The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.

In Stop mode, all the clocks in the core domain are stopped and the PLLs, HSI, HSI48 and HSE oscillators are disabled. However, the HSI can be switched ON to allow some peripherals to generate a wake-up interrupt. The LSI and LSE remain active in Stop mode.

In Standby mode, all the clocks in the core domain are stopped and the PLLs, HSI, HSI48 and HSE oscillators are disabled.

The CPU deep-sleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bit in the DBGMCU_CR register.

11.6 RCC security and privilege functional description

11.6.1 TrustZone® security protection modes

When the TrustZone® security is activated by the TZEN option byte in the flash memory option byte configuration register, the RCC is able to secure the RCC configuration and status bits from being modified by nonsecure accesses.

This is configured through the *RCC secure configuration register (RCC_SECCFGR)* to prevent nonsecure accesses to read or modify the following features:

- HSE, HSE-CSS, HSI, LSI, LSE, LSE-CSS, HSI48 configuration and status bits
- PLL1, PLL2, AHB, and APB prescalers configuration and status bits
- System clock (SYSCLK) and ICLK source clock selection and status bits
- MCO clock output configuration
- Remove reset flag RMVF configuration

If SPRIV is set in the *RCC privilege configuration register (RCC_PRIVCFGR)*, the RCC_SECCFGR register can be written only by secure and privileged accesses. If SPRIV is cleared, RCC_SECCFGR can be written only by secure accesses, privileged or unprivileged.

RCC_SECCFGR can be read by secure, nonsecure, privileged, and unprivileged accesses.

When a peripheral is configured as secure, its related clock, reset, clock source selection, and clock enable during low-power modes control bits, are also secure in the RCC_AHBxENR, RCC_APBxENR, RCC_CCIPRx and *RCC RTC domain control register (RCC_RTCCR)* registers.

A peripheral is secure when:

- For peripherals securable by TZSC (see *Section 5.4.3: TrustZone security controller (TZSC)*), the SEC security bit corresponding to this peripheral is set in the GTZC TZSC secure configuration registers.
- For TrustZone®-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 129 summarizes the RCC secured bits following the security configuration bit in the RCC_SECCFGR register.

When one security configuration bit is set, some configuration and status bits are secured. The RCC registers may contain secure and nonsecure bits:

- Secure bits: read and write operations are only allowed by secure accesses. Nonsecure read returns 0 and write accesses are ignored. No illegal access event is generated.
- Nonsecure bits: no restriction. Read and write operations are allowed by both secure and nonsecure accesses.
- A nonsecure write access to RCC_SECCFGR is ignored and generates an illegal access event. An illegal access interrupt is generated if the RCC illegal access interrupt is enabled in the GTZC TZIC registers. RCC_SECCFGR can be read by secure or nonsecure accesses.

When the TrustZone® security is disabled (TZEN = 0xC3), all registers are nonsecure. RCC_SECCFGR write accesses are ignored.

Table 129. RCC security configuration summary

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
HSISEC	HSION, HSIKERON, HSIRDY	RCC_CR
	HSICAL[11:0], HSITRIM[6:0]	RCC_HSICFGR
	HSIRDYIE	RCC_IER
	HSIRDYIF	RCC_IFR
	HSIRDYC	RCC_ICR
	CALCKEN	RCC_CCIPR1
HSESEC	HSEON, HSERDY, HSEBYP, HSECSSON, HSEEXT	RCC_CR
	HSERDYIE, HSECSSF	RCC_IER
	HSERDYIF, HSECSSF	RCC_IFR
	HSERDYC, HSECSSC	RCC_ICR
LSISEC	LSION, LSIRDY, LSIPREDIV	RCC_RTCCR
	LSIRDYIE	RCC_IER
	LSIRDYIF	RCC_IFR
	LSIRDYC	RCC_ICR
LSESEC	LSECSSON, LSECSSD, LSEDRV[1:0], LSEBYP, LSERDY, LSEON	RCC_RTCCR
	LSERDYIE, LSECSSIE	RCC_IER
	LSERDYF, LSECSSF	RCC_IFR
	LSERDYC, LSECSSC	RCC_ICR
SYSCLKSEC	SYSSW[1:0], SYSSWS[1:0], MCO1SEL[3:0], MCO1PRE[2:0], MCO2SEL[3:0], MCO2PRE[2:0]	RCC_CFGR1
	SYSTICKSEL[1:0]	RCC_CCIPR3
	VOS[1:0]	PWR_VOSR
PRESCSEC	HPRE[3:0], PPRE1[2:0], PPRE2[2:0], PPRE3[2:0]	RCC_CFGR2
PLL1SEC	PLL1SRC[1:0], PLL1RGE[1:0], PLL1FRACEN, PLL1M[3:0], PLL1VCOSEL, PLL1PEN, PLL1QEN, PLL1REN, ICLK1DIV	RCC_PLL1CFGR
	PLL1N[8:0], PLL1P[6:0], PLL1Q[6:0], PLL1R[6:0]	RCC_PLL1DIVR
	PLL1FRACN[12:0]	RCC_PLL1FRACR
	PLL1RDY, PLL1ON	RCC_CR
	PLL1RDYIE	RCC_IER
	PLL1RDYF	RCC_IFR
	PLL1RDYC	RCC_ICR

Table 129. RCC security configuration summary (continued)

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
PLL2SEC (1)	PLL2SRC[1:0], PLL2RGE[1:0], PLL2FRACEN, PLL2M[3:0], PLL2PEN, PLL2QEN, PLL2REN, PLL2VCOSEL, ICLK2DIV	RCC_PLL2CFGR
	PLL2N[8:0], PLL2P[6:0], PLL2Q[6:0], PLL2R[6:0]	RCC_PLL2DIVR
	PLL2FRACN[12:0]	RCC_PLL2FRACR
	PLL2RDY, PLL2ON	RCC_CR
	PLL2RDYIE	RCC_IER
	PLL2RDYF	RCC_IFR
	PLL2RDYC	RCC_ICR
HSI48SEC(2) (3)	HSI48ON, HSI48RDY	RCC_CR
	HSI48CAL[9:0]	RCC_CRRCR
	HSI48RDYE	RCC_IER
	HSI48RDYF	RCC_IFR
	HSI48RDYC	RCC_ICR
RMVFSEC	RMVF	RCC_RSR

1. PLL2 is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
2. TRIM field of the HSI48 is located in CRS peripheral. Be sure to secure it using CRSSEC bit in GTZC1 TZSC secure configuration register 1.
3. HSI48 is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

11.6.2 Privileged protection modes

By default, after reset, all RCC registers can be read or written by both privileged and unprivileged accesses, except the registers that can be written with privileged accesses only. *RCC privilege configuration register (RCC_PRIVCFGR)* can be read by secure and nonsecure, privileged, and unprivileged accesses.

The SPRIV bit in RCC_PRIVCFGR can be written with secure privileged accesses only. This bit configures the privileged access of all RCC secure functions (as defined by *RCC secure configuration register (RCC_SECCFGR)*, by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone®-aware peripherals).

When the SPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC secure bits is possible only with privileged accesses, including RCC_SECCFGR.
- The RCC secure bits can be read only with privileged accesses except RCC_SECCFGR and RCC_PRIVCFGR that can be read by privileged or unprivileged accesses.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit in RCC_PRIVCFGR can be written with privileged accesses only, secure or nonsecure. This bit configures the privileged access of all RCC nonsecure functions (as defined by RCC_SECCFGR, or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone®-aware peripherals).

When the NSPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC nonsecure bits is possible only with privileged accesses.
- The RCC nonsecure bits can be read only with privileged accesses except RCC_PRIVCFGR that can be read by privileged or unprivileged accesses.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and any write to these bits is ignored (RAZ/WI).

11.6.3 Clock protection

The RCC does not allow to disable the system clock (**sys_ck**). Several protections are implemented to prevent actions accidentally disabling the system clock.

In addition, PLLs and oscillators are also protected to avoid wrong programming sequences.

The protections consist in ignoring the invalid write operations except for SYSSW. Read back the protected registers to ensure that the clock configuration expected by the software is accepted by the RCC.

Table 130 shows the protections handled by the RCC.

Table 130. Clock protection summary

Protected fields	Conditions checked by the RCC
SYSSW[2:0]	The RCC prevents accidental disabling of the system clock. If SYSSW position is changed to an input having no clock available, the SYSSW remains in its original position. When the clock on the new selected position is activated and ready the SYSSW switched to the new position. It is not possible to disable the clock currently used by the SYSSW. The SYSSWS[1:0] status flag enables the application to check the selected input.
PLLxSRC[1:0]	Changing PLLxSRC value is allowed if: – The targeted clock is enabled and ready, and, – PLLx is disabled
HSEON or HSION	Changing these bits is allowed if the oscillator is not used directly or indirectly (via PLL1) as system clock
PLLxM[5:0], PLLxRGE[1:0], PLLxVCOSEL, PLLxN[8:0], PLLxP[6:0], PLLxQ[6:0], PLLxR[6:0]	Changing these fields is allowed only if the corresponding PLLx is disabled (PLLxON = 0).
PLL1ON	Changing this bit is allowed if SYSSW switch is not selecting the PLL1
PLL1PEN	Changing this bit is allowed if SYSSW switch is not selecting the PLL1
HSEBYP, HSEEXT	Changing these fields is allowed only if HSEON = 0.
LSEBYP	Changing these fields is allowed only if LSEON = 0.

A clock protection mechanism also exists in case of HSE failure (refer to *Section : Clock security system on HSE*).

11.7 RCC interrupts

The RCC provides several interrupt lines:

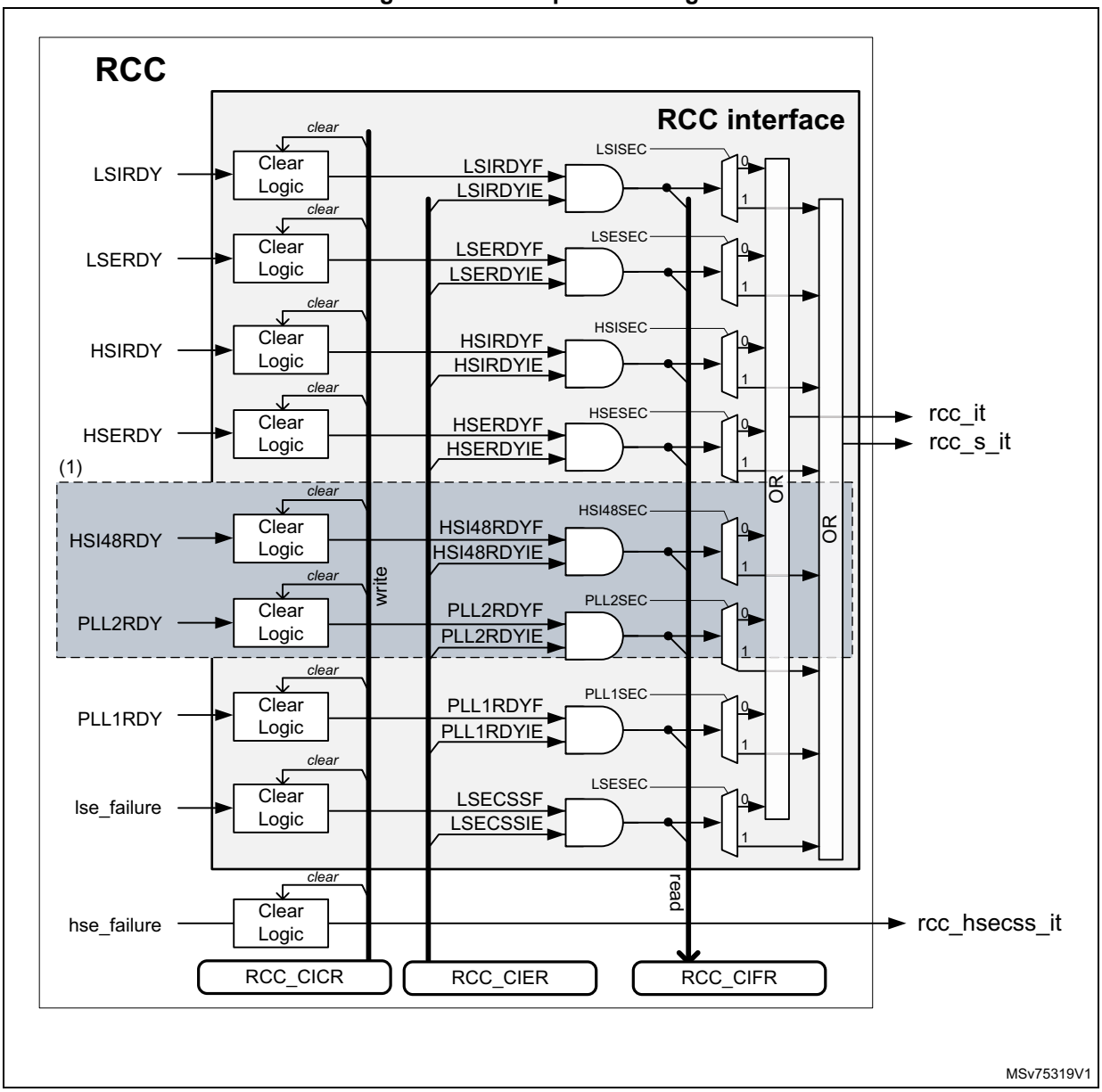
- rcc_it: general nonsecure interrupt line
- rcc_s_it: general secure interrupt line
- rcc_hsecss_it: interrupt line dedicated to the failure detection of the HSE CSS (clock security system). This line is connected to the NMI of the NVIC. The security level of this interrupt is defined by the BFHFNMINS bit of the AIRCR register, and must be aligned with the HSE security level.

The interrupt enable bits are controlled via the *RCC interrupt enable register (RCC_IER)*, except for the HSE CSS failure. When the HSE CSS feature is enabled, the generated interrupt cannot be masked.

The interrupt flags can be checked via the *RCC interrupt flag register (RCC_IFR)*. These flags can be cleared via *RCC Interrupt clear register (RCC_ICR)*.

Note: The interrupt flags are not relevant if the corresponding interrupt enable bit is not set.

Figure 77. Interrupt block diagram



1. These signals are not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

Table 131 summarizes the interrupt sources and the way to control them.

Table 131. Interrupt sources and control

Interrupt vector	Interrupt event flag	Description	Enable control bits	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby modes
RCC	LSIRDYF	LSI ready	LSIRDYIE and LSISEC = 0	Set LSIRDYC	Yes	No
	LSERDYF	LSE ready	LSERDYIE and LSESEC = 0	Set LSERDYC	Yes	No
	HSIDRYF	HSI ready	HSIDRYIE and HSISEC = 0	Set HSIRDYC	Yes	No
	HSERDYF	HSE ready	HSERDYIE and HSESEC = 0	Set HSERDYC	Yes	No
	LSECSSF	LSE CSS failure	LSECSSIE and LSESEC = 0	Set LSECSSC	Yes	Yes
	HSI48RDYF	HSI48 ready	HSI48RDYIE and HSI48SEC = 0	Set HSI48RDYC	Yes	No
	PLL1RDYF	PLL1 ready	PLL1RDYIE and PLL1SEC = 0	Set PLL1RDYC	Yes	No
	PLL2RDYF	PLL2 ready	PLL2RDYIE and PLL2SEC = 0	Set PLL2RDYC	Yes	No
RCC_S (1)	LSIRDYF	LSI ready	LSIRDYIE and LSISEC = 1	Set LSIRDYC	Yes	No
	LSERDYF	LSE ready	LSERDYIE and LSESEC = 1	Set LSERDYC	Yes	No
	HSIDRYF	HSI ready	HSIDRYIE and HSISEC = 1	Set HSIRDYC	Yes	No
	HSERDYF	HSE ready	HSERDYIE and HSESEC = 1	Set HSERDYC	Yes	No
	LSECSSF	LSE CSS failure	LSECSSIE and LSESEC = 1	Set LSECSSC	Yes	Yes
	HSI48RDYF (2)	HSI48 ready	HSI48RDYIE and HSI48SEC = 1	Set HSI48RDYC	Yes	No
	PLL1RDYF	PLL1 ready	PLL1RDYIE and PLL1SEC = 1	Set PLL1RDYC	Yes	No
	PLL2RDYF (3)	PLL2 ready	PLL2RDYIE and PLL2SEC = 1	Set PLL2RDYC	Yes	No
TAMP	ITAMP3F(4)	LSE CSS failure	LSECSSON and ITAMP3E(4) and ITAMP3IE(4)	Set CITAMP3F(4)	Yes	Yes
NMI	HSECSSF	HSE CSS failure	(5)	Set HSECSSC	Yes	No

1. The RCC secure interrupt vector is used only when TrustZone® is enabled.
2. HSI48 is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.
3. PLL2 is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

4. The LSE CSS failure event (LSECSSD) is connected to TAMP internal tamper 3. To get the interrupt associated to this event, the internal tamper 3 must be enabled, and the internal tamper 3 interrupt must be enabled. Refer to *Section 50: Tamper and backup registers (TAMP)* for details on the ITAMP3F, ITAMP3E, ITAMP3IE, and CITAMP3F bits.
5. This interrupt cannot be masked when the security system feature is enabled (HSECSSON = 1).

11.8 RCC registers

11.8.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0023

Reset by a system reset.

This register contains the main clock control and status bits.

This register can be accessed in secure or nonsecure mode, depending on the HSISEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON
				r	rw	r	rw				rw	rs	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HSI48RDY	HSI48ON	Res.	Res.	Res.	Res.	Res.	Res.	HSIDIVF	HSIDIV[1:0]		HSIKERON	HSIRDY	HSION
		r	rw							r	rw	rw	rw	r	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PLL2RDY**: PLL2 clock ready flag
Set by hardware to indicate that the PLL is locked.
0: PLL2 unlocked
1: PLL2 locked
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 26 **PLL2ON**: PLL2 enable
Set and cleared by software to enable PLL2.
Cleared by hardware when entering Stop or Standby mode.
0: PLL2 OFF (default after reset)
1: PLL2 ON
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 25 **PLL1RDY**: PLL1 clock ready flag
Set by hardware to indicate that the PLL1 is locked.
0: PLL1 unlocked (default after reset)
1: PLL1 locked

Bit 24 **PLL1ON**: PLL1 enable

Set and cleared by software to enable PLL1.

Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents clearing this bit if the PLL1 output is used as the system clock.

0: PLL1 OFF (default after reset)

1: PLL1 ON

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **HSEEXT**: External high-speed clock type in bypass mode

Set and cleared by software to select the external clock type (analog or digital).

The external clock must be enabled through the HSEON bit to be used by the device. The HSEEXT bit can be written only if the HSE oscillator is disabled.

0: HSE in analog mode (default after reset)

1: HSE in digital mode

Bit 19 **HSECSSON**: HSE clock security system enable

Set by software to enable clock security system on HSE.

This bit is “set only” (disabled by a system reset or when the system enters Standby mode). When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.

0: CSS on HSE OFF (clock detector OFF) (default after reset)

1: CSS on HSE ON (clock detector ON if the HSE oscillator is stable, OFF otherwise).

Bit 18 **HSEBYP**: HSE clock bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled through the HSEON bit to be used by the device.

The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE oscillator not bypassed (default after reset)

1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE clock is not ready (default after reset)

1: HSE clock is ready

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE when entering Stop or Standby mode.

This bit cannot be cleared if the HSE is used directly (via SYSSW mux) as system clock, or if the HSE is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set).

0: HSE is OFF (default after reset)

1: HSE is ON

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **HSI48RDY**: HSI48 clock ready flag

Set by hardware to indicate that the HSI48 oscillator is stable.

0: HSI48 clock is not ready (default after reset)

1: HSI48 clock is ready

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 12 **HSI48ON**: HSI48 clock enable

Set by software and cleared by software or by the hardware when the system enters to Stop or Standby mode.

0: HSI48 is OFF (default after reset)

1: HSI48 is ON

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **HSIDIVF**: HSI divider flag

Set and cleared by hardware.

As a write operation to HSIDIV has no immediate effect on the frequency, this flag indicates the current status of the HSI divider. HSIDIVF goes immediately to 0 when HSIDIV value is changed, and is set back to 1 when the output frequency matches the value programmed into HSIDIV[1:0].

0: new division ratio not yet propagated to hsi_ck, hsi_ker_ck (default after reset)

1: hsi_ck, hsi_ker_ck clock frequency reflects the new HSIDIV value (default register value when the clock setting is completed).

Bits 4:3 **HSIDIV[1:0]**: HSI clock divider

Set and cleared by software.

These bits select the division ratio to configure the wanted HSI clock frequency. The HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set). In that case, the new HSIDIV value is ignored.

00: division by 1, hsi_ck, hsi_ker_ck = 64 MHz (default after reset)

01: division by 2, hsi_ck, hsi_ker_ck = 32 MHz

10: division by 4, hsi_ck, hsi_ker_ck = 16 MHz

11: division by 8, hsi_ck, hsi_ker_ck = 8 MHz

Bit 2 **HSIKERON**: HSI clock enable in Stop mode

Set and cleared by software to force the HSI to ON, even in Stop mode, to be quickly available as kernel clock for peripherals. This bit has no effect on the value of HSION. It is recommended to set this bit to 1 only when HSION is set to 1.

0: no effect on HSI (default after reset)

1: HSI is forced to ON even in Stop mode

Bit 1 **HSIRDY**: HSI clock ready flag

Set by hardware to indicate that the HSI oscillator is stable.

0: HSI clock is not ready (default after reset)

1: HSI clock is ready

Bit 0 **HSION**: HSI clock enable

Set and cleared by software.

Set by hardware to force the HSI to ON when the product exits Stop or Standby mode or in case of a HSE failure which is used as the system clock source.

This bit cannot be cleared if the HSI is used directly (via SYSSW mux) as system clock, or if the HSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).

0: HSI is OFF

1: HSI is ON (default after reset)

11.8.2 RCC HSI calibration register (RCC_HSICFGR)

Address offset: 0x010

Reset value: 0x0040 0XXX

The reset value of HSICAL depends on the settings of the flash memory option bytes. HSITRIM[6:0] is reset by a system reset.

This register is used to trim the HSI clock frequency.

This register can be accessed in secure or nonsecure mode, depending on the HSISEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSICAL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **HSITRIM[6:0]**: HSI clock trimming

Set by software to adjust the calibration.

HSITRIM[6:0] bitfield is added to the engineering option bytes loaded during the reset phase (FLASH_HSI_OPT) to form the calibration trimming value.

HSICAL = HSITRIM[6:0] + FLASH_HSI_OPT.

After a change of HSITRIM[6:0], it takes one system clock cycle before the new HSITRIM[6:0] value is updated

Note: The reset value of the bitfield is 0x40.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSICAL[11:0]**: HSI clock calibration

Set by hardware by option byte loading during system reset nreset. Adjusted by software through the trimming bits HSITRIM[6:0].

This bitfield represents the sum of engineering option byte calibration and HSITRIM[6:0] values.

11.8.3 RCC clock recovery RC register (RCC_CRRCR) [a]

Address offset: 0x014

Reset value: 0x0000 0XXX

The reset value depends on the flash option bytes setting

This register shows the calibration value of HSI48.

This register can be accessed in secure or nonsecure mode, depending on the HSI48SEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **HSI48CAL[9:0]**: Internal RC 48 MHz clock calibration
Set by hardware by option-byte loading during system reset NRESET. Read-only.

a. This register is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

11.8.4 RCC clock configuration register1 (RCC_CFGR1)

Address offset: 0x01C

Reset value: 0x0000 0000

Reset by system reset

Access: word, half-word, and byte access. One or two wait states are inserted only if the access occurs during the clock source switch.

This register contains the clock configuration bits.

This register can be accessed in secure or nonsecure mode, depending on the setting of the *RCC secure configuration register (RCC_SECCFGR)*, and the RTC security signals. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2SEL[2:0]			MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]				Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RTCPRE[5:0]						Res.	Res.	Res.	SYSSWS[1:0]		Res.	SYSSW[1:0]	
		rw	rw	rw	rw	rw	rw				r	r		rw	rw

Bits 31:29 **MCO2SEL[2:0]**: microcontroller clock output 2

Set and cleared by software. Clock source selection may generate glitches on MCO2. Configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: system clock selected (**sys_ck**) (default after reset)
001: PLL2 oscillator clock selected (**pll2_p_ck**) (1)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_p_ck**)
100: LSI clock selected (**lsi_ck**)
Others: reserved

Bits 28:25 **MCO2PRE[3:0]**: MCO2 prescaler

Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. Change this prescaler only after reset, before enabling the external oscillators and the PLLs.
0000: prescaler disabled, no clock output on MCO2 (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 24:22 **MCO1SEL[2:0]**: Microcontroller clock output 1
Set and cleared by software. Clock source selection may generate glitches on MCO1. Configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: HSI clock selected (**hsi_ck**) (default after reset)
001: LSE oscillator clock selected (**lse_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_q_ck**)
100: HSI48 clock selected (**hsi48_ker_ck**)
Others: reserved

Bits 21:18 **MCO1PRE[3:0]**: MCO1 prescaler
Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. Change this prescaler only after reset, before enabling the external oscillators and the PLLs.
0000: prescaler disabled, no clock output on MCO1 (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 17:14 Reserved, must be kept at reset value.

Bits 13:8 **RTCPRE[5:0]**: HSE division factor for RTC clock
Set and cleared by software to divide the HSE to generate a clock for RTC.

Caution: The software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.

000000: no clock (default after reset)
000001: no clock
000010: HSE/2
000011: HSE/3
000100: HSE/4
...
111110: HSE/62
111111: HSE/63

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:3 **SYSSWS[1:0]**: system clock switch status
Set and cleared by hardware to indicate which clock source is used as system clock.
00: HSI used as system clock (**hsi_ck**) (default after reset).
01: HSE used as system clock (**hse_ck**)
10: PLL1 used as system clock (**pll1_p_ck**)
Others: reserved

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SYSSW[1:0]**: system clock and trace clock switch

Set and cleared by software to select system clock and trace clock sources (**sys_ck**).

Set by hardware to:

- Force the selection of the HSI when leaving a system Stop mode
- Force the selection of the HSI in case of failure of the HSE when used directly or indirectly as system clock

00: HSI selected as system clock (**hsi_ck**) (default after reset)

01: HSE selected as system clock (**hse_ck**)

10: PLL1 selected as system clock (**pll1_p_ck**)

Others: reserved

1. This source is not available on STM32G53x/54x devices. When the source is not available, the corresponding input must not be selected. Refer to the device datasheet for availability

11.8.5 RCC CPU domain clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020

Reset value: 0x0000 0000

Reset by system reset

One of two wait states are inserted only if the access occurs during the clock source switch. From 0 to 15 wait states are inserted if the access occurs when an update of the APB or AHB prescalers values is ongoing.

This register contains the clock configuration bits for the APB and AHB buses.

This register can be accessed in secure or nonsecure mode, depending on the PRESCSEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIS	APB2DIS	APB1DIS	AHB4DIS	AHB3DIS	AHB2DIS	AHB1DIS
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PPRE3[2:0]			Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
	rw	rw	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **APB3DIS**: APB3 clock disable value

Set and cleared by software.

This bit can be set to further reduce power consumption, when none of the APB3 peripherals are used and when their clocks are disabled in the *RCC APB3 peripheral clock enable register (RCC_APB3ENR)*. When this bit is set, all the APB3 peripheral clocks are OFF.

0: APB3 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB3 clock disabled

Bit 21 **APB2DIS**: APB2 clock disable value

This bit can be set to further reduce power consumption, when none of the APB2 peripherals are used and when their clocks are disabled in the *RCC APB2 peripheral clock enable register (RCC_APB2ENR)*. When this bit is

set, all the APB2 peripheral clocks are off.

0: APB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB2 clock disabled

Bit 20 **APB1DIS**: APB1 clock disable value

This bit can be set to further reduce power consumption, when none of the APB1 peripherals (except IWDG) are used and when their clocks are disabled in *RCC APB1 peripheral clock enable register (RCC_APB1LENR)* and *RCC APB1 peripheral clock enable register (RCC_APB1HENR)*. When this bit is set, all the APB1 peripheral clocks are off, except for IWDG.

0: APB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB1 clock disabled

Bit 19 **AHB4DIS**: AHB4 clock disable

This bit can be set to further reduce power consumption, when none of the AHB4 peripherals are used, and when their clocks are disabled in the *RCC AHB4 peripheral clock enable register (RCC_AHB4ENR)*. When this bit is set, all the AHB4 peripheral clocks from RCC_AHB4ENR are OFF.

0: AHB4 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB4 clock disabled

Bit 18 **AHB3DIS**: AHB3 clock disable

This bit can be set to further reduce power consumption, when none of the AHB3 peripherals are used and when their clocks are disabled in the *RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)*. When this bit is set, all the AHB3 peripheral clocks from RCC_AHB3ENR are OFF.

0: AHB3 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB3 clock disabled

Bit 17 **AHB2DIS**: AHB2 clock disable

This bit can be set to further reduce power consumption, when none of the AHB2 peripherals are used and when their clocks are disabled in the *RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)*. When this bit is set, all the AHB2 peripheral clocks are OFF, except for SRAM2 (and SRAM3 if available in the product).

0: AHB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB2 clock disabled

Bit 16 **AHB1DIS**: AHB1 clock disable

This bit can be set to further reduce power consumption, when none of the AHB1 peripherals are used and when their clocks are disabled in the *RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)*. When this bit is set, all the AHB1 peripheral clocks are OFF, except for FLASH, ICACHE, DCACHE and SRAM1.

0: AHB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB1 clock disabled

Note: DCACHE is not always present. Refer to the device datasheet for availability.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PPRE3[2:0]**: APB low-speed prescaler (APB3)
Set and cleared by software to control the APB low-speed clock division factor.
The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE3 write.
0xx: rcc_pclk3 = rcc_hclk1
100: rcc_pclk3 = rcc_hclk1 / 2
101: rcc_pclk3 = rcc_hclk1 / 4
110: rcc_pclk3 = rcc_hclk1 / 8
111: rcc_pclk3 = rcc_hclk1 / 16

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **PPRE2[2:0]**: APB high-speed prescaler (APB2)
Set and cleared by software to control APB high-speed clocks division factor.
The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE2 write.
0xx: rcc_pclk2 = rcc_hclk1
100: rcc_pclk2 = rcc_hclk1 / 2
101: rcc_pclk2 = rcc_hclk1 / 4
110: rcc_pclk2 = rcc_hclk1 / 8
111: rcc_pclk2 = rcc_hclk1 / 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE1[2:0]**: APB low-speed prescaler (APB1)
Set and cleared by software to control the division factor of rcc_pclk1.
The clock is divided by the new prescaler factor from 1 to 16 cycles of rcc_hclk after PPRE write.
0xx: rcc_pclk1 = rcc_hclk1 (default after reset)
100: rcc_pclk1 = rcc_hclk1 / 2
101: rcc_pclk1 = rcc_hclk1 / 4
110: rcc_pclk1 = rcc_hclk1 / 8
111: rcc_pclk1 = rcc_hclk1 / 16

Bits 3:0 **HPRE[3:0]**: AHB prescaler
Set and cleared by software to control the division factor of rcc_hclk. Changing this division ratio has an impact on the frequency of all bus matrix clocks
0xxx: rcc_hclk = sys_ck (default after reset)
1000: rcc_hclk = sys_ck / 2
1001: rcc_hclk = sys_ck / 4
1010: rcc_hclk = sys_ck / 8
1011: rcc_hclk = sys_ck / 16
1100: rcc_hclk = sys_ck / 64
1101: rcc_hclk = sys_ck / 128
1110: rcc_hclk = sys_ck / 256
1111: rcc_hclk = sys_ck / 512

Caution: Be careful when using the voltage scaling. Due to the propagation delay of the new division factor, after a prescaler factor change, and before lowering the V_{CORE} voltage, this register must be read to check that the new prescaler value has been taken into account.

Depending on the clock source frequency and the voltage range, the software application must program a correct value in HPRE to make sure that the system frequency does not exceed the maximum frequency.

11.8.6 RCC PLL clock source selection register (RCC_PLL1CFGR)

Address offset: 0x028

Reset value: 0x0000 0000

Access: No wait state, word, half-word, and byte access

This register can be accessed in secure or nonsecure mode, depending on th PLL1SEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ICLK1DIV	Res.	Res.	Res.	Res.	Res.	PLL1REN	PLL1QEN	PLL1PEN
							rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PLL1M[5:0]						Res.	Res.	PLL1VCOSEL	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ICLK1DIV**: Intermediate clock 1 division

Set and cleared by software to select the intermediate clock (ic1_ck) division ratio. There is no need to disable ICLK1 to change the division ratio.

0: Intermediate clock 1 division ratio is 5 (default after reset)

1: Intermediate clock 1 division ratio is 3

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **PLL1REN**: PLL1 DIVR and ICLK1 divider output enable

Set and cleared by software to enable the pll1_r_ck and ic1_ck outputs.

To save power, PLL1REN and PLL1R[6:0] bits must be cleared when the ic1_ck is not used.

0: pll1_r_ck and ic1_ck outputs disabled (default after reset)

1: pll1_r_ck and ic1_ck outputs enabled

Bit 17 **PLL1QEN**: PLL1 DIVQ divider output enable

Set and cleared by software to enable the pll1_q_ck output of the PLL1.

To save power, when the pll1_q_ck output of the PLL1 is not used, the pll1_q_ck must be disabled.

0: pll1_q_ck output disabled (default after reset)

1: pll1_q_ck output enabled

Bit 16 **PLL1PEN**: PLL1 DIVP divider output enable

Set and cleared by software to enable the pll1_p_ck output of the PLL1. The hardware prevents clearing this bit if the PLL1 output is used as the system clock.

To save power, when the pll1_p_ck output of the PLL1 is not used, the pll1_p_ck must be disabled.

0: pll1_p_ck output disabled (default after reset)

1: pll1_p_ck output enabled

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **PLL1M[5:0]**: Prescaler for PLL1

Set and cleared by software to configure the prescaler of the PLL1.
The hardware does not allow any modification of this bitfield when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
To save power when PLL1 is not used, the value of PLL1M must be cleared.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL1VCOSEL**: PLL1 VCO selection

Set and cleared by software to select the proper VCO frequency range used for PLL1. The hardware does not allow any modification of this bitfield when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
0: wide VCO range: 128 to 560 MHz (default after reset)
1: medium VCO range: 150 to 420 MHz

Bit 4 **PLL1FRACEN**: PLL1 fractional latch enable

Set and cleared by software to latch the content of PLL1FRACN into the sigma-delta modulator.
To latch the PLL1FRACN value into the sigma-delta modulator, PLL1FRACEN must be cleared, then set. The transition 0 to 1 transfers the content of PLL1FRACN into the modulator.

Bits 3:2 **PLL1RGE[1:0]**: PLL1 input frequency range

Set and cleared by software to select the proper reference frequency range used for PLL1. The hardware does not allow any modification of this bitfield when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
00: PLL1 input (ref1_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL1 input (ref1_ck) clock range frequency between 2 and 4 MHz
10: PLL1 input (ref1_ck) clock range frequency between 4 and 8 MHz
11: PLL1 input (ref1_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL1SRC[1:0]**: PLL1M and PLLs clock source selection

Set and cleared by software to select the PLL clock source. The hardware does not allow any modification of this bitfield when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
To save power, when no PLL is used, the value of PLL1SRC must be set to 0b00.
00: no clock sent to PLL1M divider and PLL1 (default after reset).
01: HSI selected as PLL clock (hsi_ck)
10: HSE selected as PLL clock (hse_ck)
11: no clock sent to PLL1M divider and PLL1

11.8.7 RCC PLL clock source selection register (RCC_PLL2CFGR) [a]

Address offset: 0x02C

Reset value: 0x0000 0000

Access: no wait state, word, half-word, and byte access.

This register can be accessed in secure or nonsecure mode, depending on the PLL2SEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ICLK2DIV	Res.	Res.	Res.	Res.	Res.	PLL2REN	PLL2QEN	PLL2PEN
							rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PLL2M[5:0]						Res.	Res.	PLL2VCOSEL	PLL2FRACEN	PLL2RGE[1:0]		PLL2SRC[1:0]	
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ICLK2DIV**: Intermediate clock 2 division

Set and cleared by software to select the intermediate clock (ic2_ck) division ratio. There is no need to disable ICLK2 to change the division ratio.

0: Intermediate clock 2 division ratio is 5 (default after reset)

1: Intermediate clock 2 division ratio is 3

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **PLL2REN**: PLL2 DIVR and ICLK2 divider output enable

Set and cleared by software to enable the pll2_r_ck and ic2_ck outputs.

To save power, PLL2REN and PLL2R[6:0] bits must be cleared when the ic2_ck is not used.

0: pll2_r_ck and ic2_ck output disabled (default after reset)

1: pll2_r_ck and ic2_ck outputs enabled

Bit 17 **PLL2QEN**: PLL2 DIVQ divider output enable

Set and cleared by software to enable the pll2_q_ck output of the PLL2.

To save power, when the pll2_q_ck output of the PLL2 is not used, the pll2_q_ck must be disabled.

0: pll2_q_ck output disabled (default after reset)

1: pll2_q_ck output enabled

a. This register is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

Bit 16 **PLL2PEN**: PLL2 DIVP divider output enable
Set and cleared by software to enable the pll2_p_ck output of the PLL2.
To save power, when the pll2_p_ck output of the PLL2 is not used, the pll2_p_ck must be disabled.
0: pll2_p_ck output disabled (default after reset)
1: pll2_p_ck output enabled

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **PLL2M[5:0]**: prescaler for PLL2
Set and cleared by software to configure the prescaler of the PLL2.
The hardware does not allow any modification of this bitfield when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
To save power when PLL2 is not used, the value of PLL2M must be cleared.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL2VCOSEL**: PLL2 VCO selection
Set and cleared by software to select the proper VCO frequency range used for PLL2.
The hardware does not allow any modification of this bitfield when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
0: wide VCO range: 128 to 560 MHz (default after reset)
1: medium VCO range: 150 to 420 MHz

Bit 4 **PLL2FRACEN**: PLL2 fractional latch enable
Set and cleared by software to latch the content of PLL2FRACN into the sigma-delta modulator. In order to latch the PLL2FRACN value into the sigma-delta modulator, PLL2FRACEN must be cleared, then set. The transition 0 to 1 transfers the content of PLL2FRACN into the modulator.

Bits 3:2 **PLL2RGE[1:0]**: PLL2 input frequency range
Set and cleared by software to select the proper reference frequency range used for PLL2.
The hardware does not allow any modification of this bitfield when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
00: PLL2 input (ref2_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL2 input (ref2_ck) clock range frequency between 2 and 4 MHz
10: PLL2 input (ref2_ck) clock range frequency between 4 and 8 MHz
11: PLL2 input (ref2_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL2SRC[1:0]**: PLL2 clock source selection
Set and cleared by software to select the PLL clock source.
The hardware does not allow any modification of this bitfield when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
To save power, when no PLL is used, the value of PLL2SRC must be set to 0b00.
00: no clock send to PLL2M divider and PLL2 (default after reset)
01: HSI selected as PLL2 clock (hsi_ck)
10: HSE selected as PLL2 clock (hse_ck)
11: no clock send to PLL2M divider and PLL2

11.8.8 RCC PLL1 divider register (RCC_PLL1DIVR)

Address offset: 0x034

Reset value: 0x0101 0280

Access: no wait state, word, half-word, and byte access.

This register can be accessed in secure or nonsecure mode, depending on the PLL1SEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL1R[6:0]							Res.	PLL1Q[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1P[6:0]								PLL1N[8:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **PLL1R[6:0]**: PLL1 DIVR division factor

Set and cleared by software to control the frequency of the pll1_r_ck clock.
The hardware does not allow any modification of this bitfield when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
0000000: pll1_r_ck = vco1_ck / 1
0000001: pll1_r_ck = vco1_ck / 2 (default after reset)
0000010: pll1_r_ck = vco1_ck / 3
0000011: pll1_r_ck = vco1_ck / 4
...
1111111: pll1_r_ck = vco1_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL1Q[6:0]**: PLL1 DIVQ division factor

Set and cleared by software to control the frequency of the pll1_q_ck clock.
The hardware does not allow any modification of this bitfield when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
0000000: pll1_q_ck = vco1_ck
0000001: pll1_q_ck = vco1_ck / 2 (default after reset)
0000010: pll1_q_ck = vco1_ck / 3
0000011: pll1_q_ck = vco1_ck / 4
...
1111111: pll1_q_ck = vco1_ck / 128

Bits 15:9 **PLL1P[6:0]**: PLL1 DIVP division factor

Set and cleared by software to control the frequency of the pll1_p_ck clock.
The hardware does not allow any modification of this bitfield when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
Note that odd division factors are not allowed.
0000000: Not allowed
0000001: pll1_p_ck = vco1_ck / 2 (default after reset)
0000010: Not allowed
0000011: pll1_p_ck = vco1_ck / 4
...
1111111: pll1_p_ck = vco1_ck / 128

Bits 8:0 **PLL1N[8:0]**: Multiplication factor for PLL1VCO

Set and cleared by software to control the multiplication factor of the VCO.
The hardware does not allow any modification of this bitfield when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
0x003: PLL1N = 4
0x004: PLL1N = 5
0x005: PLL1N = 6
...
0x080: PLL1N = 129 (default after reset)
...
0x1FF: PLL1N = 512
Others: reserved

Caution: The software must set correctly these bits to insure that the VCO output frequency is in the valid frequency range, that is:
128 to 560 MHz if PLL1VCOSEL = 0
150 to 420 MHz if PLL1VCOSEL = 1
VCO output frequency = F_{ref1_ck} x PLL1N, with PLL1N between 4 and 512

11.8.9 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state, word, and half-word access.

This register can be accessed in secure or nonsecure mode, depending on the PLL1SEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL1FRACN[12:0]**: Fractional part of the multiplication factor for PLL1 VCO

Set and cleared by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

VCO output frequency = F_{ref1_ck} × (PLL1N + (PLL1FRACN / 2^{13})), with

- PLL1N between 4 and 512
- PLL1FRACN between 0 and 2^{13} - 1

To change the PLL1FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

1. Clear the bit PLL1FRACEN.
2. Write the new fractional value into PLL1FRACN.
3. Set the bit PLL1FRACEN.

Bits 2:0 Reserved, must be kept at reset value.

11.8.10 RCC PLL2 divider register (RCC_PLL2DIVR) [a]

Address offset: 0x03C

Reset value: 0x0101 0280

Access: no wait state, word, half-word, and byte access.

This register can be accessed in secure or nonsecure mode, depending on the PLL2SEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL2R[6:0]							Res.	PLL2Q[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2P[6:0]								PLL2N[8:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **PLL2R[6:0]**: PLL2 DIVR division factor

Set and cleared by software to control the frequency of the pll2_r_ck clock.
The hardware does not allow any modification of this bitfield when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
0000000: pll2_r_ck = vco2_ck
0000001: pll2_r_ck = vco2_ck / 2 (default after reset)
0000010: pll2_r_ck = vco2_ck / 3
0000011: pll2_r_ck = vco2_ck / 4
...
1111111: pll2_r_ck = vco2_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL2Q[6:0]**: PLL2 DIVQ division factor

Set and cleared by software to control the frequency of the pll2_q_ck clock.
The hardware does not allow any modification of this bitfield when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
0000000: pll2_q_ck = vco2_ck
0000001: pll2_q_ck = vco2_ck / 2 (default after reset)
0000010: pll2_q_ck = vco2_ck / 3
0000011: pll2_q_ck = vco2_ck / 4
...
1111111: pll2_q_ck = vco2_ck / 128

a. This register is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

Bits 15:9 **PLL2P[6:0]**: PLL2 DIVP division factor

Set and cleared by software to control the frequency of the pll2_p_ck clock.
The hardware does not allow any modification of this bitfield when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
0000000: pll2_p_ck = vco2_ck
0000001: pll2_p_ck = vco2_ck / 2 (default after reset)
0000010: pll2_p_ck = vco2_ck / 3
0000011: pll2_p_ck = vco2_ck / 4
...
1111111: pll2_p_ck = vco2_ck / 128

Bits 8:0 **PLL2N[8:0]**: Multiplication factor for PLL2VCO

Set and cleared by software to control the multiplication factor of the VCO.
The hardware does not allow any modification of this bitfield when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
0x003: PLL2N = 4
0x004: PLL2N = 5
0x005: PLL2N = 6
...
0x080: PLL2N = 129 (default after reset)
...
0x1FF: PLL2N = 512
Others: reserved

Caution: The software must set correctly these bits to ensure that the VCO output frequency is in the valid frequency range, that is:
128 to 560 MHz if PLL2VCOSEL = 0
150 to 420 MHz if PLL2VCOSEL = 1
VCO output frequency = F_{ref2_ck} x PLL2N, with PLL2N between 4 and 512

11.8.11 RCC PLL2 fractional divider register (RCC_PLL2FRACR) [a]

Address offset: 0x040

Reset value: 0x0000 0000

Access: no wait state; word and half-word access.

This register can be accessed in secure or nonsecure mode, depending on the PLL2SEC bit of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL2FRACN[12:0]:** fractional part of the multiplication factor for PLL2 VCO

Set and cleared by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO.

VCO output frequency = F_{ref2_ck} x (PLL2N + (PLL2FRACN / 2^{13})), with

- PLL2N between 4 and 512
- PLL2FRACN between 0 and 2^{13} - 1

To change the PLL2FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

1. Clear the bit PLL2FRACEN.
2. Write the new fractional value into PLL2FRACN.
3. Set the bit PLL2FRACEN.

Bits 2:0 Reserved, must be kept at reset value.

a. This register is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

11.8.12 RCC interrupt enable register (RCC_IER)

Address offset: 0x050

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security bits of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSECSSIE	Res.	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSIRDYIE
						rw		rw	rw	rw	rw	rw		rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **LSECSSIE**: LSE clock security system interrupt enable

Set and cleared by software to enable/disable the interrupt caused by the clock security system (CSS) on the LSE oscillator.

0: LSE CSS interrupt disabled (default after reset)

1: LSE CSS interrupt enabled

Bit 8 Reserved, must be kept at reset value.

Bit 7 **PLL2RDYIE**: PLL2 ready interrupt enable

Set and cleared by software to enable/disable the interrupt caused by the PLL2 lock.

0: PLL2 lock interrupt disabled (default after reset)

1: PLL2 lock interrupt enabled

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 **PLL1RDYIE**: PLL1 ready interrupt enable

Set and cleared by software to enable/disable the interrupt caused by the PLL1 lock.

0: PLL1 lock interrupt disabled (default after reset)

1: PLL1 lock interrupt enabled

Bit 5 **HSI48RDYIE: HSI48 ready interrupt enable**

Set and cleared by software to enable/disable the interrupt caused by the HSI48 oscillator stabilization.

0: HSI48 ready interrupt disabled (default after reset)

1: HSI48 ready interrupt enabled

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 4 **HSERDYIE**: HSE ready interrupt enable

Set and cleared by software to enable/disable the interrupt caused by the HSE oscillator stabilization.

0: HSE ready interrupt disabled (default after reset)

1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI ready interrupt enable

Set and cleared by software to enable/disable the interrupt caused by the HSI oscillator stabilization.

0: HSI ready interrupt disabled (default after reset)

1: HSI ready interrupt enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYIE**: LSE ready interrupt enable

Set and cleared by software to enable/disable the interrupt caused by the LSE oscillator stabilization.

0: LSE ready interrupt disabled (default after reset)

1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable

Set and cleared by software to enable/disable the interrupt caused by the LSI oscillator stabilization.

0: LSI ready interrupt disabled (default after reset)

1: LSI ready interrupt enabled

11.8.13 RCC interrupt flag register (RCC_IFR)

Address offset: 0x054

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security bits of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSF	LSECSSF	Res.	PLL2RDYF	PLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	Res.	LSERDYF	LSIRDYF
					r	r		r	r	r	r	r		r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSF**: HSE clock security system interrupt flag
Cleared by software by writing HSECSSC bit.
Set by hardware in case of HSE clock failure.
0: no clock security interrupt caused by HSE clock failure (default after reset)
1: clock security interrupt caused by HSE clock failure

Bit 9 **LSECSSF**: LSE clock security system interrupt flag
Cleared by software by writing LSECSSC bit.
Set by hardware in case of LSE clock failure if LSECSSIE is set.
0: no clock security interrupt caused by HSE clock failure (default after reset)
1: clock security interrupt caused by HSE clock failure

Bit 8 Reserved, must be kept at reset value.

Bit 7 **PLL2RDYF**: PLL2 ready interrupt flag
Cleared by software by writing PLL2RDYC bit.
Set by hardware when the PLL2 locks and PLL2RDYIE is set.
0: no clock ready interrupt caused by PLL2 lock (default after reset)
1: clock ready interrupt caused by PLL2 lock

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 **PLL1RDYF**: PLL1 ready interrupt flag
Cleared by software by writing PLL1RDYC bit.
Set by hardware when the PLL1 locks and PLL1RDYIE is set.
0: no clock ready interrupt caused by PLL1 lock (default after reset)
1: clock ready interrupt caused by PLL1 lock

Bit 5 **HSI48RDYF**: HSI48 ready interrupt flag
Cleared by software by writing HSI48RDYC bit.
Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
0: no clock ready interrupt caused by the HSI48 oscillator (default after reset)
1: clock ready interrupt caused by the HSI48 oscillator
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 4 **HSERDYF**: HSE ready interrupt flag
Cleared by software by writing HSERDYC bit.
Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
0: no clock ready interrupt caused by the HSE (default after reset)
1: clock ready interrupt caused by the HSE

Bit 3 **HSIRDYF**: HSI ready interrupt flag
Cleared by software by writing HSIRDYC bit.
Set by hardware when the HSI clock becomes stable and HSIRDYIE is set.
0: no clock ready interrupt caused by the HSI (default after reset)
1: clock ready interrupt caused by the HSI

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYF**: LSE ready interrupt flag
Cleared by software by writing LSERDYC bit.
Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
0: no clock ready interrupt caused by the LSE (default after reset)
1: clock ready interrupt caused by the LSE

Bit 0 **LSIRDYF**: LSI ready interrupt flag
Cleared by software by writing LSIRDYC bit.
Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.
0: no clock ready interrupt caused by the LSI (default after reset)
1: clock ready interrupt caused by the LSI

11.8.14 RCC Interrupt clear register (RCC_ICR)

Address offset: 0x058

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security bits of the *RCC secure configuration register (RCC_SECCFGR)*. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSC	LSECSSC	Res.	PLL2RDYC	PLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	Res.	LSERDYC	LSIRDYC
					rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSC**: HSE clock security system interrupt clear
Set by software to clear HSECSSF.
Cleared by hardware when clear done.
0: HSECSSF no effect (default after reset)
1: HSECSSF cleared

Bit 9 **LSECSSC**: LSE clock security system interrupt clear
Set by software to clear LSECSSF.
Cleared by hardware when clear done.
0: LSECSSF no effect (default after reset)
1: LSECSSF cleared

Bit 8 Reserved, must be kept at reset value.

Bit 7 **PLL2RDYC**: PLL2 ready interrupt clear
Set by software to clear PLL2RDYF.
Cleared by hardware when clear done.
0: PLL2RDYF no effect (default after reset)
1: PLL2RDYF cleared

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear
Set by software to clear PLL1RDYF.
Cleared by hardware when clear done.
0: PLL1RDYF no effect (default after reset)
1: PLL1RDYF cleared

Bit 5 **HSI48RDYC**: HSI48 ready interrupt clear
Set by software to clear HSI48RDYF.
Cleared by hardware when clear done.
0: HSI48RDYF no effect (default after reset)
1: HSI48RDYF cleared
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 4 **HSERDYC**: HSE ready interrupt clear
Set by software to clear HSERDYF.
Cleared by hardware when clear done.
0: HSERDYF no effect (default after reset)
1: HSERDYF cleared

Bit 3 **HSIRDYC**: HSI ready interrupt clear
Set by software to clear HSIRDYF.
Cleared by hardware when clear done.
0: HSIRDYF no effect (default after reset)
1: HSIRDYF cleared

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYC**: LSE ready interrupt clear
Set by software to clear LSERDYF.
Cleared by hardware when clear done.
0: LSERDYF no effect (default after reset)
1: LSERDYF cleared

Bit 0 **LSIRDYC**: LSI ready interrupt clear
Set by software to clear LSIRDYF.
Cleared by hardware when clear done.
0: LSIRDYF no effect (default after reset)
1: LSIRDYF cleared

11.8.15 RCC AHB1 reset register (RCC_AHB1RSTR)

Address offset: 0x060

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGRST	Res.
														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC1RST	CORDIC1RST	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2RST	GPDMA1RST
rw	rw		rw											rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGRST**: RAMCFG block reset
Set and cleared by software.
0: does not reset RAMCFG block (default after reset)
1: resets RAMCFG block

Bit 16 Reserved, must be kept at reset value.

Bit 15 **FMAC1RST**: FMAC1 block reset
Set and cleared by software.
0: does not reset FMAC1 block (default after reset)
1: resets FMAC1 block

Bit 14 **CORDIC1RST**: CORDIC1 block reset
Set and cleared by software.
0: does not reset CORDIC1 block (default after reset)
1: resets CORDIC1 block

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC block reset Set and cleared by software.
0: does not reset CRC block (default after reset)
1: resets CRC block

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **GPDMA2RST**: GPDMA2 block reset
Set and cleared by software.
0: does not reset GPDMA2 block (default after reset)
1: resets GPDMA2 block

Bit 0 **GPDMA1RST**: GPDMA1 block reset
Set and cleared by software.
0: does not reset GPDMA1 block (default after reset)
1: resets GPDMA1 block

11.8.16 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x064

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3RST	ADC12RST
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC2RST	Res.	Res.	Res.	Res.	Res.	DAC10RST	DAC9RST	DAC8RST	DAC7RST	DAC6RST	DAC5RST	Res.	Res.	DAC2RST	DAC1RST
rw						rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **ADC3RST**: ADC3 blocks reset
Set and cleared by software.
0: does not reset ADC3 blocks (default after reset)
1: resets ADC3 blocks

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 16 **ADC12RST**: ADC1 and 2 blocks reset
Set and cleared by software.
0: does not reset ADC1 and 2 blocks (default after reset)
1: resets ADC1 and 2 blocks

Bit 15 **FMAC2RST**: FMAC2 block reset
Set and cleared by software.
0: does not reset FMAC2 block (default after reset)
1: resets FMAC2 block

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **DAC10RST**: DAC10 block reset
Set and cleared by software.
0: does not reset DAC10 block (default after reset)
1: resets DAC10 block

Bit 8 **DAC9RST**: DAC9 block reset
Set and cleared by software.
0: does not reset DAC9 block (default after reset)
1: resets DAC9 block

Bit 7 **DAC8RST**: DAC8 block reset
Set and cleared by software.
0: does not reset DAC8 block (default after reset)
1: resets DAC8 block

Bit 6 **DAC7RST**: DAC7 block reset
Set and cleared by software.
0: does not reset DAC7 block (default after reset)
1: resets DAC7 block

Bit 5 **DAC6RST**: DAC6 block reset
Set and cleared by software.
0: does not reset DAC6 block (default after reset)
1: resets DAC6 block

Bit 4 **DAC5RST**: DAC5 block reset
Set and cleared by software.
0: does not reset DAC5 block (default after reset)
1: resets DAC5 block

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DAC2RST**: DAC2 block reset
Set and cleared by software.
0: does not reset DAC2 block (default after reset)
1: resets DAC2 block

Bit 0 **DAC1RST**: DAC1 block reset
Set and cleared by software.
0: does not reset DAC1 block (default after reset)
1: resets DAC1 block

11.8.17 RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

Address offset: 0x06C

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1RST	Res.	Res.	Res.	GPIOHRST
											rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST	Res.	OTFDEC1RST	Res.	CCBRST	SAESRST	PKARST	RNGRST	HASHRST	AESRST
rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **XSPI1RST**: XSPI1 block reset
Set and cleared by software.
0: does not reset XSPI1 block (default after reset)
1: resets XSPI1 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **GPIOHRST**: GPIOH block reset
Set and cleared by software.
0: does not reset the GPIOH block (default after reset)
1: resets the GPIOH block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 15 **GPIOGRST**: GPIOG block reset
Set and cleared by software.
0: does not reset the GPIOG block (default after reset)
1: resets the GPIOG block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **GPIOFRST**: GPIOF block reset
Set and cleared by software.
0: does not reset the GPIOF block (default after reset)
1: resets the GPIOF block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 13 **GPIOERST**: GPIOE block reset
Set and cleared by software.
0: does not reset the GPIOE block (default after reset)
1: resets the GPIOE block

Bit 12 **GPIODRST**: GPIOD block reset
Set and cleared by software.
0: does not reset the GPIOD block (default after reset)
1: resets the GPIOD block

Bit 11 **GPIOCRST**: GPIOC block reset
Set and cleared by software.
0: does not reset the GPIOC block (default after reset)
1: resets the GPIOC block

Bit 10 **GPIOBRST**: GPIOB block reset
Set and cleared by software.
0: does not reset the GPIOB block (default after reset)
1: resets the GPIOB block

Bit 9 **GPIOARST**: GPIOA block reset
Set and cleared by software.
0: does not reset the GPIOA block (default after reset)
1: resets the GPIOA block

Bit 8 Reserved, must be kept at reset value.

Bit 7 **OTFDEC1RST**: OTFDEC1 block reset
Set and cleared by software.
0: does not reset OTFDEC1 block (default after reset)
1: resets OTFDEC1 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CCBRST**: CCB block reset
Set and cleared by software.
0: does not reset CCB block (default after reset)
1: resets CCB block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 4 **SAESRST**: SAES block reset
Set and cleared by software.
0: does not reset SAES block (default after reset)
1: resets SAES block
Note: This bit is not available on STM32G53x/54x/55x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 3 **PKARST**: PKA block reset
Set and cleared by software.
0: does not reset PKA block (default after reset)
1: resets PKA block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 2 **RNGRST**: RNG block reset
Set and cleared by software.
0: does not reset RNG block (default after reset)
1: resets RNG block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 1 **HASHRST**: HASH block reset
Set and cleared by software.
0: does not reset HASH block (default after reset)
1: resets HASH block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 0 **AESRST**: AES block reset
Set and cleared by software.
0: does not reset AES block (default after reset)
1: resets AES block
Note: This bit is not available on STM32G53x/55x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

11.8.18 RCC APB1 peripheral low reset register (RCC_APB1LRSTR)

Address offset: 0x074

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	UART6RST	CRSRST	I3C1RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.
						rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3RST	SPI2RST	Res.	Res.	TIM14RST	TIM13RST	TIM12RST	TIM11RST	TIM10RST	TIM9RST	TIM7RST	TIM6RST	Res.	Res.	TIM3RST	TIM2RST
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **UART6RST**: UART6 block reset
Set and cleared by software.
0: does not reset the UART6 block (default after reset)
1: resets the UART6 block

Bit 24 **CRSRST**: CRS block reset
Set and cleared by software.
0: does not reset the CRS block (default after reset)
1: resets the CRS block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 23 **I3C1RST**: I3C1 block reset
Set and cleared by software.
0: does not reset the I3C1 block (default after reset)
1: resets the I3C1 block

Bit 22 **I2C2RST**: I2C2 block reset
Set and cleared by software.
0: does not reset the I2C2 block (default after reset)
1: resets the I2C2 block

Bit 21 **I2C1RST**: I2C1 block reset
Set and cleared by software.
0: does not reset the I2C1 block (default after reset)
1: resets the I2C1 block

Bit 20 **UART5RST**: UART5 block reset
Set and cleared by software.
0: does not reset the UART5 block (default after reset)
1: resets the UART5 block

Bit 19 **UART4RST**: UART4 block reset
Set and cleared by software.
0: does not reset the UART4 block (default after reset)
1: resets the UART4 block

Bit 18 **USART3RST**: USART3 block reset
Set and cleared by software.
0: does not reset the USART3 block (default after reset)
1: resets the USART3 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 17 **USART2RST**: USART2 block reset
Set and cleared by software.
0: does not reset the USART2 block (default after reset)
1: resets the USART2 block

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3RST**: SPI3 block reset
Set and cleared by software.
0: does not reset the SPI3 block (default after reset)
1: resets the SPI3 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **SPI2RST**: SPI2 block reset
Set and cleared by software.
0: does not reset the SPI2 block (default after reset)
1: resets the SPI2 block

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **TIM14RST**: TIM14 block reset
Set and cleared by software.
0: does not reset the TIM14 block (default after reset)
1: resets the TIM14 block

Bit 10 **TIM13RST**: TIM13 block reset
Set and cleared by software.
0: does not reset the TIM13 block (default after reset)
1: resets the TIM13 block

Bit 9 **TIM12RST**: TIM12 block reset
Set and cleared by software.
0: does not reset the TIM12 block (default after reset)
1: resets the TIM12 block

Bit 8 **TIM11RST**: TIM12 block reset
Set and cleared by software.
0: does not reset the TIM12 block (default after reset)
1: resets the TIM12 block

Bit 7 **TIM10RST**: TIM12 block reset
Set and cleared by software.
0: does not reset the TIM12 block (default after reset)
1: resets the TIM12 block

Bit 6 **TIM9RST**: TIM12 block reset
Set and cleared by software.
0: does not reset the TIM12 block (default after reset)
1: resets the TIM12 block

Bit 5 **TIM7RST**: TIM7 block reset
Set and cleared by software.
0: does not reset the TIM7 block (default after reset)
1: resets the TIM7 block

Bit 4 **TIM6RST**: TIM6 block reset
Set and cleared by software.
0: does not reset the TIM6 block (default after reset)
1: resets the TIM6 block

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3RST**: TIM3 block reset
Set and cleared by software.
0: does not reset the TIM3 block (default after reset)
1: resets the TIM3 block

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 0 **TIM2RST**: TIM2 block reset
Set and cleared by software.
0: does not reset the TIM2 block (default after reset)
1: resets the TIM2 block

11.8.19 RCC APB1 peripheral high reset register (RCC_APB1HRSTR)

Address offset: 0x078

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.
								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	Res.	Res.	Res.	DTSRST	Res.	Res.	Res.
						rw						rw			

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UCPD1RST**: UCPD1 block reset
Set and cleared by software.
0: does not reset the UCPD1 block (default after reset)
1: resets the UCPD1 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FDCANRST**: FDCAN1, FDCAN2 and FDCAN3 block reset
Set and cleared by software.
0: does not reset the FDCAN1, FDCAN2 and FDCAN3 blocks (default after reset)
1: resets the FDCAN1, FDCAN2 and FDCAN3 blocks
Note: This bit is not available on STM32G53x/54x devices. When the bit is not available the corresponding input must not be selected. Refer to the device datasheet for availability.
Some STM32G56x/55x/54x/53x devices only have one FDCAN instance. Refer to the device datasheet for availability.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **DTSRST**: DTS block reset
Set and cleared by software.
0: does not reset the DTS block (default after reset)
1: resets the DTS block

Bits 2:0 Reserved, must be kept at reset value.

11.8.20 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x07C

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIM5RST	TIM4RST	Res.	Res.	Res.	Res.	USBRST	Res.	Res.	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST
	rw	rw					rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLAY1APPRST	PLAY1RST	Res.	HRTIM1RST
	rw	rw	rw	rw								rw	rw		rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TIM5RST**: TIM5 block reset
Set and cleared by software.
0: does not reset the TIM5 block (default after reset)
1: resets the TIM5 block

Bit 29 **TIM4RST**: TIM4 block reset
Set and cleared by software.
0: does not reset the TIM4 block (default after reset)
1: resets the TIM4 block

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **USBRST**: USB block reset
Set and cleared by software.
0: does not reset the USB block (default after reset)
1: resets the USB block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST**: TIM17 block reset
Set and cleared by software.
0: does not reset the TIM17 block (default after reset)
1: resets the TIM17 block

Bit 17 **TIM16RST**: TIM16 block reset
Set and cleared by software.
0: does not reset the TIM16 block (default after reset)
1: resets the TIM16 block

Bit 16 **TIM15RST**: TIM15 block reset
Set and cleared by software.
0: does not reset the TIM15 block (default after reset)
1: resets the TIM15 block

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 block reset
Set and cleared by software.
0: does not reset the USART1 block (default after reset)
1: resets the USART1 block

Bit 13 **TIM8RST**: TIM8 block reset
Set and cleared by software.
0: does not reset the TIM8 block (default after reset)
1: resets the TIM8 block

Bit 12 **SPI1RST**: SPI1 block reset
Set and cleared by software.
0: does not reset the SPI1 block (default after reset)
1: resets the SPI1 block

Bit 11 **TIM1RST**: TIM1 block reset
Set and cleared by software.
0: does not reset the TIM1 block (default after reset)
1: resets the TIM1 block

Bits 10:4 Reserved, must be kept at reset value.

Bit 3 **PLAY1APPRST**: PLAY1 application block reset
Set and cleared by software. This bit reset the PLAY1 but not its configuration. When the reset is released the PLAY1 restarts.
0: does not reset PLAY1 block (default after reset)
1: resets PLAY1 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 2 **PLAY1RST**: PLAY1 global block reset
Set and cleared by software. This bit resets the PLAY1 block including its configuration.
0: does not reset PLAY1 block (default after reset)
1: resets PLAY1 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **HRTIM1RST**: HRTIM1 block reset
Set and cleared by software.
0: does not reset the HRTIM1 block (default after reset)
1: resets the HRTIM1 block

11.8.21 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x080

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP8RST	COMP7RST	COMP6RST	COMP5RST	COMP4RST	COMP3RST	COMP2RST	COMP1RST	Res.	VREFRST	Res.	Res.	Res.	OPAMP3RST	OPAMP2RST	OPAMP1RST
rw	rw	rw	rw	rw	rw	rw	rw		rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POSENC2RST	POSENC1RST	Res.	Res.	LPTIM1RST	Res.	Res.	I2C4RST	I2C3RST	LPUART1RST	Res.	TIM27RST	TIM26RST	TIM20RST	Res.	Res.
rw	rw			rw			rw	rw	rw		rw	rw	rw		

Bit 31 **COMP8RST**: COMP8 block reset
Set and cleared by software.
0: does not reset the COMP8 block (default after reset)
1: resets the COMP8 block

Bit 30 **COMP7RST**: COMP7 block reset
Set and cleared by software.
0: does not reset the COMP7 block (default after reset)
1: resets the COMP7 block

Bit 29 **COMP6RST**: COMP6 block reset
Set and cleared by software.
0: does not reset the COMP6 block (default after reset)
1: resets the COMP6 block

Bit 28 **COMP5RST**: COMP5 block reset
Set and cleared by software.
0: does not reset the COMP5 block (default after reset)
1: resets the COMP5 block

Bit 27 **COMP4RST**: COMP4 block reset
Set and cleared by software.
0: does not reset the COMP4 block (default after reset)
1: resets the COMP4 block

Bit 26 **COMP3RST**: COMP3 block reset
Set and cleared by software.
0: does not reset the COMP3 block (default after reset)
1: resets the COMP3 block

Bit 25 **COMP2RST**: COMP2 block reset
Set and cleared by software.
0: does not reset the COMP2 block (default after reset)
1: resets the COMP2 block

Bit 24 **COMP1RST**: COMP1 block reset
Set and cleared by software.
0: does not reset the COMP1 block (default after reset)
1: resets the COMP1 block

Bit 23 Reserved, must be kept at reset value.

Bit 22 **VREFRST**: VREFBUF block reset
Set and cleared by software.
0: does not reset the VREFBUF block (default after reset)
1: resets the VREFBUF block

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **OPAMP3RST**: OPAMP3 block reset
Set and cleared by software.
0: does not reset the OPAMP3 block (default after reset)
1: resets the OPAMP3 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 17 **OPAMP2RST**: OPAMP2 block reset
Set and cleared by software.
0: does not reset the OPAMP2 block (default after reset)
1: resets the OPAMP2 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 16 **OPAMP1RST**: OPAMP1 block reset
Set and cleared by software.
0: does not reset the OPAMP1 block (default after reset)
1: resets the OPAMP1 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 15 **POSENC2RST**: POSENC2 block reset
Set and cleared by software.
0: does not reset the POSENC2 block (default after reset)
1: resets the POSENC2 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **POSENC1RST**: POSENC1 block reset
Set and cleared by software.
0: does not reset the POSENC1 block (default after reset)
1: resets the POSENC1 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1RST**: LPTIM1 block reset
Set and cleared by software.
0: does not reset the LPTIM1 block (default after reset)
1: resets the LPTIM1 block

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **I2C4RST**: I2C4 block reset
Set and cleared by software.
0: does not reset the I2C4 block (default after reset)
1: resets the I2C4 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 7 **I2C3RST**: I2C3 block reset
Set and cleared by software.
0: does not reset the I2C3 block (default after reset)
1: resets the I2C3 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 **LPUART1RST**: LPUART1 block reset
Set and cleared by software.
0: does not reset the LPUART1 block (default after reset)
1: resets the LPUART1 block

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TIM27RST**: TIM27 block reset
Set and cleared by software.
0: does not reset the TIM27 block (default after reset)
1: resets the TIM27 block

Bit 3 **TIM26RST**: TIM26 block reset
Set and cleared by software.
0: does not reset the TIM26 block (default after reset)
1: resets the TIM26 block

Bit 2 **TIM20RST**: TIM20 block reset
Set and cleared by software.
0: does not reset the TIM20 block (default after reset)
1: resets the TIM20 block
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 1:0 Reserved, must be kept at reset value.

11.8.22 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x088

Reset value: 0xCC00 0100

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1EN	DCACHEEN	Res.	Res.	SRAM3EN	SRAM2EN	Res.	TZSC1EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	Res.
rw	rw			rw	rw		rw							rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC1EN	CORDIC1EN	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2EN	GPDMA1EN
rw	rw		rw				rw							rw	rw

Bit 31 **SRAM1EN**:SRAM1 clock enable
Set and cleared by software.
0: SRAM1 clock disabled
1: SRAM1 clock enabled (default after reset)

Bit 30 **DCACHEEN**: DCACHE clock enable
Set and cleared by software
0: DCACHE peripheral clock disabled
1: DCACHE peripheral clock enabled (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SRAM3EN**:SRAM3 clock enable
Set and cleared by software.
0: SRAM3 clock disabled
1: SRAM3 clock enabled (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 26 **SRAM2EN**:SRAM2 clock enable
Set and cleared by software.
0: SRAM2 clock disabled
1: SRAM2 clock enabled (default after reset)

Bit 25 Reserved, must be kept at reset value.

Bit 24 **TZSC1EN**: TZSC1 clock enable
Set and cleared by software
0: TZSC1 peripheral clock disabled (default after reset)
1: TZSC1 peripheral clock enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGEN**: RAMCFG clock enable
Set and cleared by software.
0: RAMCFG peripheral clock disabled (default after reset)
1: RAMCFG peripheral clock enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **FMAC1EN**: FMAC1 clock enable
Set and cleared by software.
0: FMAC1 peripheral clock disabled (default after reset)
1: FMAC1 peripheral clock enabled

Bit 14 **CORDIC1EN**: CORDIC1 clock enable
Set and cleared by software.
0: CORDIC1 peripheral clock disabled (default after reset)
1: CORDIC1 peripheral clock enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable
Set and cleared by software.
0: CRC peripheral clock disabled (default after reset)
1: CRC peripheral clock enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHEN**: Flash memory interface clock enable
Set and cleared by software.
0: Flash memory interface clock disabled
1: Flash memory interface clock enabled (default after reset)

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **GPDMA2EN**: GPDMA2 clock enable
Set and cleared by software.
0: GPDMA2 peripheral clock disabled (default after reset)
1: GPDMA2 peripheral clock enabled

Bit 0 **GPDMA1EN**: GPDMA1 clock enable
Set and cleared by software.
0: GPDMA1 peripheral clock disabled (default after reset)
1: GPDMA1 peripheral clock enabled

11.8.23 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x08C

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3EN	ADC12EN
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC2EN	Res.	Res.	Res.	Res.	Res.	DAC10EN	DAC9EN	DAC8EN	DAC7EN	DAC6EN	DAC5EN	Res.	Res.	DAC2EN	DAC1EN
rw						rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **ADC3EN**: ADC3 peripheral clocks enable
Set and cleared by software.
0: ADC3 peripheral clocks disabled (default after reset)
1: ADC3 peripheral clocks enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 16 **ADC12EN**: ADC1 and ADC2 peripheral clocks enable
Set and cleared by software.
0: ADC1 and ADC2 peripheral clocks disabled (default after reset)
1: ADC1 and ADC2 peripheral clocks enabled

Bit 15 **FMAC2EN**: FMAC2 clock enable
Set and cleared by software.
0: FMAC2 peripheral clock disabled (default after reset)
1: FMAC2 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **DAC10EN**: DAC10 clock enable
Set and cleared by software.
0: DAC10 peripheral clock disabled (default after reset)
1: DAC10 peripheral clock enabled

Bit 8 **DAC9EN**: DAC9 clock enable
Set and cleared by software.
0: DAC9 peripheral clock disabled (default after reset)
1: DAC9 peripheral clock enabled

Bit 7 **DAC8EN**: DAC8 clock enable
Set and cleared by software.
0: DAC8 peripheral clock disabled (default after reset)
1: DAC8 peripheral clock enabled

Bit 6 **DAC7EN**: DAC7 clock enable
Set and cleared by software.
0: DAC7 peripheral clock disabled (default after reset)
1: DAC7 peripheral clock enabled

Bit 5 **DAC6EN**: DAC6 clock enable
Set and cleared by software.
0: DAC6 peripheral clock disabled (default after reset)
1: DAC6 peripheral clock enabled

Bit 4 **DAC5EN**: DAC5 clock enable
Set and cleared by software.
0: DAC5 peripheral clock disabled (default after reset)
1: DAC5 peripheral clock enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DAC2EN**: DAC2 clock enable
Set and cleared by software.
0: DAC2 peripheral clock disabled (default after reset)
1: DAC2 peripheral clock enabled

Bit 0 **DAC1EN**: DAC1 clock enable
Set and cleared by software.
0: DAC1 peripheral clock disabled (default after reset)
1: DAC1 peripheral clock enabled

11.8.24 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)

Address offset: 0x090

Reset value: 0x0000 0000

This register has no security or privilege protection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCKEN	Res.	Res.	Res.	Res.	Res.
										rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **DBGCKEN**: DBG clock enable
Set and cleared by software.
0: DBG clock disabled (default after reset)
1: DBG clock enabled

Bits 20:0 Reserved, must be kept at reset value.

11.8.25 RCC AHB4 peripheral clock enable register (RCC_AHB4ENR)

Address offset: 0x094

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1EN	Res.	Res.	Res.	GPIOHEN
											rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN	Res.	OTFDEC1EN	Res.	CCBEN	SAESEN	PKAEN	RNGEN	HASHEN	AESEN
rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **XSPI1EN**: XSPI1 clock enable
Set and cleared by software.
0: XSPI1 peripheral clock disabled (default after reset)
1: XSPI1 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **GPIOHEN**: GPIOH clock enable
Set and cleared by software.
0: GPIOH peripheral clock disabled (default after reset)
1: GPIOH peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 15 **GPIOGEN**: GPIOG clock enable
Set and cleared by software.
0: GPIOG peripheral clock disabled (default after reset)
1: GPIOG peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **GPIOFEN**: GPIOF clock enable
Set and cleared by software.
0: GPIOF peripheral clock disabled (default after reset)
1: GPIOF peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 13 **GPIOEEN**: GPIOE clock enable
Set and cleared by software.
0: GPIOE peripheral clock disabled (default after reset)
1: GPIOE peripheral clock enabled

Bit 12 **GPIODEN**: GPIOD clock enable
Set and cleared by software.
0: GPIOD peripheral clock disabled (default after reset)
1: GPIOD peripheral clock enabled

Bit 11 **GPIOCEN**: GPIOC clock enable
Set and cleared by software.
0: GPIOC peripheral clock disabled (default after reset)
1: GPIOC peripheral clock enabled

Bit 10 **GPIOBEN**: GPIOB clock enable
Set and cleared by software.
0: GPIOB peripheral clock disabled (default after reset)
1: GPIOB peripheral clock enabled

Bit 9 **GPIOAEN**: GPIOA clock enable
Set and cleared by software.
0: GPIOA peripheral clock disabled (default after reset)
1: GPIOA peripheral clock enabled

Bit 8 Reserved, must be kept at reset value.

Bit 7 **OTFDEC1EN**: OTFDEC1 clock enable
Set and cleared by software.
0: OTFDEC1 peripheral clock disabled (default after reset)
1: OTFDEC1 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CCBEN**: CCB clock enable
Set and cleared by software.
0: CCB peripheral clock disabled (default after reset)
1: CCB peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 4 **SAESEN**: SAES clock enable
Set and cleared by software.
0: SAES peripheral clock disabled (default after reset)
1: SAES peripheral clock enabled
Note: This bit is not available on STM32G53x/54x/55x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 3 **PKAEN**: PKA clock enable
Set and cleared by software.
0: PKA peripheral clock disabled (default after reset)
1: PKA peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 2 **RNGEN**: RNG clock enable
Set and cleared by software.
0: RNG peripheral clock disabled (default after reset)
1: RNG peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 1 **HASHEN**: HASH clock enable
Set and cleared by software.
0: HASH peripheral clock disabled (default after reset)
1: HASH peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 0 **AESEN**: AES clock enable
Set and cleared by software.
0: AES peripheral clock disabled (default after reset)
1: AES peripheral clock enabled
Note: This bit is not available on STM32G53x/55x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

11.8.26 RCC APB1 peripheral clock enable register (RCC_APB1LENR)

Address offset: 0x09C

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	UART6EN	CRSEN	I3C1EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.
						rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	Res.	WWDGEN	TIM14EN	TIM13EN	TIM12EN	TIM11EN	TIM10EN	TIM9EN	TIM7EN	TIM6EN	Res.	Res.	TIM3EN	TIM2EN
rw	rw		rs	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **UART6EN**: UART6 clock enable
Set and cleared by software.
0: UART6 peripheral clock disabled (default after reset)
1: UART6 peripheral clock enabled

Bit 24 **CRSEN**: CRS clock enable
Set and cleared by software.
0: CRS peripheral clock disabled (default after reset)
1: CRS peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 23 **I3C1EN**: I3C1 clock enable
Set and cleared by software.
0: I3C1 peripheral clock disabled (default after reset)
1: I3C1 peripheral clock enabled

Bit 22 **I2C2EN**: I2C2 clock enable
Set and cleared by software.
0: I2C2 peripheral clock disabled (default after reset)
1: I2C2 peripheral clock enabled

Bit 21 **I2C1EN**: I2C1 clock enable
Set and cleared by software.
0: I2C1 peripheral clock disabled (default after reset)
1: I2C1 peripheral clock enabled

Bit 20 **UART5EN**: UART5 clock enable
Set and cleared by software.
0: UART5 peripheral clock disabled (default after reset)
1: UART5 peripheral clock enabled

Bit 19 **UART4EN**: UART4 clock enable
Set and cleared by software.
0: UART4 peripheral clock disabled (default after reset)
1: UART4 peripheral clock enabled

Bit 18 **USART3EN**: USART3 clock enable
Set and cleared by software.
0: USART3 peripheral clock disabled (default after reset)
1: USART3 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 17 **USART2EN**: USART2 clock enable
Set and cleared by software.
0: USART2 peripheral clock disabled (default after reset)
1: USART2 peripheral clock enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3EN**: SPI3 clock enable
Set and cleared by software.
0: SPI3 peripheral clock disabled (default after reset)
1: SPI3 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **SPI2EN**: SPI2 clock enable
Set and cleared by software.
0: SPI2 peripheral clock disabled (default after reset)
1: SPI2 peripheral clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **WWDGEN**: WWDG clock enable
This bit is set by software to enable the window watchdog clock. It is reset by hardware system reset. This bit can also be set by hardware if the WWDG_SW option bit is reset.
0: WWDG peripheral clock disabled (default after reset)
1: WWDG peripheral clock enabled

Bit 11 **TIM14EN**: TIM14 clock enable
Set and cleared by software.
0: TIM14 peripheral clock disabled (default after reset)
1: TIM14 peripheral clock enabled

Bit 10 **TIM13EN**: TIM13 clock enable
Set and cleared by software.
0: TIM13 peripheral clock disabled (default after reset)
1: TIM13 peripheral clock enabled

Bit 9 **TIM12EN**: TIM12 clock enable
Set and cleared by software.
0: TIM12 peripheral clock disabled (default after reset)
1: TIM12 peripheral clock enabled

Bit 8 **TIM11EN**: TIM11 clock enable
Set and cleared by software.
0: TIM11 peripheral clock disabled (default after reset)
1: TIM11 peripheral clock enabled

Bit 7 **TIM10EN**: TIM10 clock enable
Set and cleared by software.
0: TIM10 peripheral clock disabled (default after reset)
1: TIM10 peripheral clock enabled

Bit 6 **TIM9EN**: TIM9 clock enable
Set and cleared by software.
0: TIM9 peripheral clock disabled (default after reset)
1: TIM9 peripheral clock enabled

Bit 5 **TIM7EN**: TIM7 clock enable
Set and cleared by software.
0: TIM7 peripheral clock disabled (default after reset)
1: TIM7 peripheral clock enabled

Bit 4 **TIM6EN**: TIM6 clock enable
Set and cleared by software.
0: TIM6 peripheral clock disabled (default after reset)
1: TIM6 peripheral clock enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3EN**: TIM3 clock enable
Set and cleared by software.
0: TIM3 peripheral clock disabled (default after reset)
1: TIM3 peripheral clock enabled

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 0 **TIM2EN**: TIM2 clock enable
Set and cleared by software.
0: TIM2 peripheral clock disabled (default after reset)
1: TIM2 peripheral clock enabled

11.8.27 RCC APB1 peripheral clock enable register (RCC_APB1HENR)

Address offset: 0x0A0

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.
								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	Res.	Res.	Res.	DTSEN	Res.	Res.	Res.
						rw						rw			

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UCPD1EN**: UCPD1 clock enable
Set and cleared by software.
0: UCPD1 peripheral clock disabled (default after reset)
1: UCPD1 peripheral clock enabled

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FDCANEN**: FDCAN1, FDCAN2 and FDCAN3 peripheral clock enable
Set and cleared by software.
0: FDCAN1, FDCAN2 and FDCAN3 peripheral clock disabled (default after reset)
1: FDCAN1, FDCAN2 and FDCAN3 peripheral clock enabled

Note: Some STM32G56x/55x/54x/53x devices only have one FDCAN instance. Refer to the device datasheet for availability.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **DTSEN**: DTS clock enable
Set and cleared by software.
0: DTS peripheral clock disabled (default after reset)
1: DTS peripheral clock enabled

Bits 2:0 Reserved, must be kept at reset value.

11.8.28 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x0A4

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIM5EN	TIM4EN	Res.	Res.	Res.	Res.	USBEN	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN
	rw	rw					rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLAY1EN	PLAY1APBEN	Res.	HRTIM1EN
	rw	rw	rw	rw								rw	rw		rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TIM5EN**: TIM5 clock enable
Set and cleared by software.
0: TIM5 peripheral clock disabled (default after reset)
1: TIM5 peripheral clock enabled

Bit 29 **TIM4EN**: TIM4 clock enable
Set and cleared by software.
0: TIM4 peripheral clock disabled (default after reset)
1: TIM4 peripheral clock enabled

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **USBEN**: USB clock enable
Set and cleared by software.
0: USB peripheral clock disabled (default after reset)
1: USB peripheral clock enabled

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM17 clock enable
Set and cleared by software.
0: TIM17 peripheral clock disabled (default after reset)
1: TIM17 peripheral clock enabled

Bit 17 **TIM16EN**: TIM16 clock enable
Set and cleared by software.
0: TIM16 peripheral clock disabled (default after reset)
1: TIM16 peripheral clock enabled

Bit 16 **TIM15EN**: TIM15 clock enable
Set and cleared by software.
0: TIM15 peripheral clock disabled (default after reset)
1: TIM15 peripheral clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1 clock enable
Set and cleared by software.
0: USART1 peripheral clock disabled (default after reset)
1: USART1 peripheral clock enabled

Bit 13 **TIM8EN**: TIM8 clock enable
Set and cleared by software.
0: TIM8 peripheral clock disabled (default after reset)
1: TIM8 peripheral clock enabled

Bit 12 **SPI1EN**: SPI1 clock enable
Set and cleared by software.
0: SPI1 peripheral clock disabled (default after reset)
1: SPI1 peripheral clock enabled

Bit 11 **TIM1EN**: TIM1 clock enable
Set and cleared by software.
0: TIM1 peripheral clock disabled (default after reset)
1: TIM1 peripheral clock enabled

Bits 10:4 Reserved, must be kept at reset value.

Bit 3 **PLAY1EN**: PLAY1 clock enable
Set and cleared by software.
0: PLAY1 peripheral clock disabled (default after reset)
1: PLAY1 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 2 **PLAY1APBEN**: PLAY1 APB clock enable
Set and cleared by software.
0: PLAY1 APB clock disabled (default after reset)
1: PLAY1 APB clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **HRTIM1EN**: HRTIM1 clock enable
Set and cleared by software.
0: HRTIM1 peripheral clock disabled (default after reset)
1: HRTIM1 peripheral clock enabled

11.8.29 RCC APB3 peripheral clock enable register (RCC_APB3ENR)

Address offset: 0x0A8

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP8EN	COMP7EN	COMP6EN	COMP5EN	COMP4EN	COMP3EN	COMP2EN	COMP1EN	RTCAPBEN	VREFEN	Res.	Res.	Res.	OPAMP3EN	OPAMP2EN	OPAMP1EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POSENC2EN	POSENC1EN	Res.	Res.	LPTIM1EN	Res.	Res.	I2C4EN	I2C3EN	LPUART1EN	Res.	TIM27EN	TIM26EN	TIM20EN	SBSEN	Res.
rw	rw			rw			rw	rw	rw		rw	rw	rw	rw	

Bit 31 **COMP8EN**: COMP8 clock enable
Set and cleared by software.
0: COMP8 clock disabled (default after reset)
1: COMP8 clock enabled

Bit 30 **COMP7EN**: COMP7 clock enable
Set and cleared by software.
0: COMP7 clock disabled (default after reset)
1: COMP7 clock enabled

Bit 29 **COMP6EN**: COMP6 clock enable
Set and cleared by software.
0: COMP6 clock disabled (default after reset)
1: COMP6 clock enabled

Bit 28 **COMP5EN**: COMP5 clock enable
Set and cleared by software.
0: COMP5 clock disabled (default after reset)
1: COMP5 clock enabled

Bit 27 **COMP4EN**: COMP4 clock enable
Set and cleared by software.
0: COMP4 clock disabled (default after reset)
1: COMP4 clock enabled

Bit 26 **COMP3EN**: COMP3 clock enable
Set and cleared by software.
0: COMP3 clock disabled (default after reset)
1: COMP3 clock enabled

Bit 25 **COMP2EN**: COMP2 clock enable
Set and cleared by software.
0: COMP2 clock disabled (default after reset)
1: COMP2 clock enabled

Bit 24 **COMP1EN**: COMP1 clock enable
Set and cleared by software.
0: COMP1 clock disabled (default after reset)
1: COMP1 clock enabled

Bit 23 **RTCAPBEN**: RTC APB interface clock enable
Set and cleared by software.
0: RTC APB interface clock disabled (default after reset)
1: RTC APB interface clock enabled

Bit 22 **VREFEN**: VREFBUF clock enable
Set and cleared by software.
0: VREFBUF peripheral clock disabled (default after reset)
1: VREFBUF peripheral clock enabled

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **OPAMP3EN**: OPAMP3 clock enable
Set and cleared by software.
0: OPAMP3 clock disabled (default after reset)
1: OPAMP3 clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 17 **OPAMP2EN**: OPAMP2 clock enable
Set and cleared by software.
0: OPAMP2 clock disabled (default after reset)
1: OPAMP2 clock enabled
This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 16 **OPAMP1EN**: OPAMP1 clock enable
Set and cleared by software.
0: OPAMP1 clock disabled (default after reset)
1: OPAMP1 clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 15 **POSENC2EN**: POSENC2 clock enable
Set and cleared by software.
0: POSENC2 clock disabled (default after reset)
1: POSENC2 clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **POSENC1EN**: POSENC1 clock enable
Set and cleared by software.
0: POSENC1 clock disabled (default after reset)
1: POSENC1 clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1EN**: LPTIM1 clock enable
Set and cleared by software.
0: LPTIM1 peripheral clock disabled (default after reset)
1: LPTIM1 peripheral clock enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **I2C4EN**: I2C4 clock enable
Set and cleared by software.
0: I2C4 peripheral clock disabled (default after reset)
1: I2C4 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 7 **I2C3EN**: I2C3 clock enable
Set and cleared by software.
0: I2C3 peripheral clock disabled (default after reset)
1: I2C3 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 **LPUART1EN**: LPUART1 clock enable
Set and cleared by software.
0: LPUART1 peripheral clock disabled (default after reset)
1: LPUART1 peripheral clock enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TIM27EN**: TIM27 clock enable
Set and cleared by software.
0: TIM27 peripheral clock disabled (default after reset)
1: TIM27 peripheral clock enabled

Bit 3 **TIM26EN**: TIM26 clock enable
Set and cleared by software.
0: TIM26 peripheral clock disabled (default after reset)
1: TIM26 peripheral clock enabled

Bit 2 **TIM20EN**: TIM20 clock enable
Set and cleared by software.
0: TIM20 peripheral clock disabled (default after reset)
1: TIM20 peripheral clock enabled
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 1 **SBSEN**: SBS clock enable
Set and cleared by software.
0: SBS peripheral clock disabled (default after reset)
1: SBS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.

11.8.30 RCC AHB1 peripheral low-power clock enable register (RCC_AHB1LPENR)

Address offset: 0x0B0

Reset value: 0xED02 D103

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1LPEN	DCACHELPEN	ICACHELPEN	Res.	SRAM3LPEN	SRAM2LPEN	Res.	TZSC1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGLPEN	Res.
rw	rw	rw		rw	rw		rw							rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC1LPEN	CORDIC1LPEN	Res.	CRCLPEN	Res.	Res.	Res.	FLASHLPEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2LPEN	GPDMA1LPEN
rw	rw		rw				rw							rw	rw

Bit 31 **SRAM1LPEN**: SRAM1 clock enable in low-power mode
Set and cleared by software
0: SRAM1 clock disabled in low-power mode
1: SRAM1 clock enabled in low-power mode (default after reset)

Bit 30 **DCACHELPEN**: DCACHE clock enable in low-power mode
Set and cleared by software
0: DCACHE peripheral clock disabled in low-power mode
1: DCACHE peripheral clock enabled in low-power mode (default after reset)

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 29 **ICACHELPEN**: ICACHE clock enable in low-power mode
Set and cleared by software
0: ICACHE peripheral clock disabled in low-power mode
1: ICACHE peripheral clock enabled in low-power mode (default after reset)

Bit 28 Reserved, must be kept at reset value.

Bit 27 **SRAM3LPEN**: SRAM3 clock enable in low-power mode
Set and cleared by software
0: SRAM3 peripheral clock disabled in low-power mode
1: SRAM3 peripheral clock enabled in low-power mode (default after reset)

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 26 **SRAM2LPEN**: SRAM2 clock enable in low-power mode
Set and cleared by software
0: SRAM2 peripheral clock disabled in low-power mode
1: SRAM2 peripheral clock enabled in low-power mode (default after reset)

Bit 25 Reserved, must be kept at reset value.

Bit 24 **TZSC1LPEN**: TZSC1 clock enable in low-power mode
Set and cleared by software
0: TZSC1 peripheral clock disabled in low-power mode
1: TZSC1 peripheral clock enabled in low-power mode (default after reset)

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGLPEN**: RAMCFG clock enable in low-power mode
Set and cleared by software.
0: RAMCFG peripheral clock disabled in low-power mode
1: RAMCFG peripheral clock enabled in low-power mode (default after reset)

Bit 16 Reserved, must be kept at reset value.

Bit 15 **FMAC1LPEN**: FMAC1 clock enable in low-power mode
Set and cleared by software.
0: FMAC1 peripheral clock disabled in low-power mode
1: FMAC1 peripheral clock enabled in low-power mode (default after reset)

Bit 14 **CORDIC1LPEN**: CORDIC1 clock enable in low-power mode
Set and cleared by software.
0: CORDIC1 peripheral clock disabled in low-power mode
1: CORDIC1 peripheral clock enabled in low-power mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCLPEN**: CRC clock enable in low-power mode
Set and cleared by software.
0: CRC peripheral clock disabled in low-power mode
1: CRC peripheral clock enabled in low-power mode (default after reset)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHLPEN**: Flash memory interface clock enable in low-power mode
Set and cleared by software.
0: Flash memory interface clock disabled in low-power mode
1: Flash memory interface clock enabled in low-power mode (default after reset)

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **GPDMA2LPEN**: GPDMA2 clock enable in low-power mode
Set and cleared by software.
0: GPDMA2 peripheral clock disabled in low-power mode
1: GPDMA2 peripheral clock enabled in low-power mode (default after reset)

Bit 0 **GPDMA1LPEN**: GPDMA1 clock enable in low-power mode
Set and cleared by software.
0: GPDMA1 peripheral clock disabled in low-power mode
1: GPDMA1 peripheral clock enabled in low-power mode (default after reset)

11.8.31 RCC AHB2 peripheral low-power clock enable register (RCC_AHB2LPENR)

Address offset: 0x0B4

Reset value: 0x0003 83F3

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3LPEN	ADC12LPEN
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC2LPEN	Res.	Res.	Res.	Res.	Res.	DAC10LPEN	DAC9LPEN	DAC8LPEN	DAC7LPEN	DAC6LPEN	DAC5LPEN	Res.	Res.	DAC2LPEN	DAC1LPEN
rw						rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **ADC3LPEN**: ADC3 peripheral clock enable in low-power mode
Set and cleared by software.
0: ADC3 peripheral clocks disabled in low-power mode
1: ADC3 peripheral clocks enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 16 **ADC12LPEN**: ADC1 and 2 peripheral clock enable in low-power mode
Set and cleared by software.
0: ADC1 and 2 peripheral clock disabled in low-power mode
1: ADC1 and 2 peripheral clock enabled in low-power mode (default after reset)

Bit 15 **FMAC2LPEN**: FMAC2 clock enable in low-power mode
Set and cleared by software.
0: FMAC2 peripheral clock disabled in low-power mode
1: FMAC2 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **DAC10LPEN**: DAC10 clock enable in low-power mode
Set and cleared by software.
0: DAC10 peripheral clock disabled in low-power mode
1: DAC10 peripheral clock enabled in low-power mode (default after reset)

Bit 8 **DAC9LPEN**: DAC9 clock enable in low-power mode
Set and cleared by software.
0: DAC9 peripheral clock disabled in low-power mode
1: DAC9 peripheral clock enabled in low-power mode (default after reset)

Bit 7 **DAC8LPEN**: DAC8 clock enable in low-power mode
Set and cleared by software.
0: DAC8 peripheral clock disabled in low-power mode
1: DAC8 peripheral clock enabled in low-power mode (default after reset)

Bit 6 **DAC7LPEN**: DAC7 clock enable in low-power mode
Set and cleared by software.
0: DAC7 peripheral clock disabled in low-power mode
1: DAC7 peripheral clock enabled in low-power mode (default after reset)

Bit 5 **DAC6LPEN**: DAC6 clock enable in low-power mode
Set and cleared by software.
0: DAC6 peripheral clock disabled in low-power mode
1: DAC6 peripheral clock enabled in low-power mode (default after reset)

Bit 4 **DAC5LPEN**: DAC5 clock enable in low-power mode
Set and cleared by software.
0: DAC5 peripheral clock disabled in low-power mode
1: DAC5 peripheral clock enabled in low-power mode (default after reset)

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DAC2LPEN**: DAC2 clock enable in low-power mode
Set and cleared by software.
0: DAC2 peripheral clock disabled in low-power mode
1: DAC2 peripheral clock enabled in low-power mode (default after reset)

Bit 0 **DAC1LPEN**: DAC1 clock enable in low-power mode
Set and cleared by software.
0: DAC1 peripheral clock disabled in low-power mode
1: DAC1 peripheral clock enabled in low-power mode (default after reset)

11.8.32 RCC AHB3 peripheral low-power clock enable register (RCC_AHB3LPENR)

Address offset: 0x0B8

Reset value: 0x0020 0000

This register has no security or privilege protection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCKLPEN	Res.	Res.	Res.	Res.	Res.
										rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **DBGCKLPEN**: DBG clock enable in low-power mode
Set and cleared by software.
0: DBG clock disabled in low-power mode
1: DBG clock enabled in low-power mode (default after reset)

Bits 20:0 Reserved, must be kept at reset value.

11.8.33 RCC AHB4 peripheral low-power clock enable register (RCC_AHB4LPENR)

Address offset: 0x0BC

Reset value: 0x0011 FEBF

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1LPEN	Res.	Res.	Res.	GPIOHLPEN
											rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN	Res.	OTFDEC1LPEN	Res.	CCBLPEN	SAESLPEN	PKALPEN	RNGLPEN	HASHLPEN	AESLPEN
rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **XSPI1LPEN**: XSPI1 clock enable in low-power mode
Set and cleared by software.
0: XSPI1 peripheral clock disabled in low-power mode
1: XSPI1 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **GPIOHLPEN**: GPIOH clock enable in low-power mode
Set and cleared by software.
0: GPIOH peripheral clock disabled in low-power mode
1: GPIOH peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 15 **GPIOGLPEN**: GPIOG clock enable in low-power mode
Set and cleared by software.
0: GPIOG peripheral clock disabled in low-power mode
1: GPIOG peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **GPIOFLPEN**: GPIOF clock enable in low-power mode
Set and cleared by software.
0: GPIOF peripheral clock disabled in low-power mode
1: GPIOF peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 13 **GPIOELPEN**: GPIOE clock enable in low-power mode
Set and cleared by software.
0: GPIOE peripheral clock disabled in low-power mode
1: GPIOE peripheral clock enabled in low-power mode (default after reset)

Bit 12 **GPIODLPEN**: GPIOD clock enable in low-power mode
Set and cleared by software.
0: GPIOD peripheral clock disabled in low-power mode
1: GPIOD peripheral clock enabled in low-power mode (default after reset)

Bit 11 **GPIOCLPEN**: GPIOC clock enable in low-power mode
Set and cleared by software.
0: GPIOC peripheral clock disabled in low-power mode
1: GPIOC peripheral clock enabled in low-power mode (default after reset)

Bit 10 **GPIOBLPEN**: GPIOB clock enable in low-power mode
Set and cleared by software.
0: GPIOB peripheral clock disabled in low-power mode
1: GPIOB peripheral clock enabled in low-power mode (default after reset)

Bit 9 **GPIOALPEN**: GPIOA clock enable in low-power mode
Set and cleared by software.
0: GPIOA peripheral clock disabled in low-power mode
1: GPIOA peripheral clock enabled in low-power mode (default after reset)

Bit 8 Reserved, must be kept at reset value.

Bit 7 **OTFDEC1LPEN**: OTFDEC1 clock enable in low-power mode
Set and cleared by software.
0: OTFDEC1 peripheral clock disabled in low-power mode
1: OTFDEC1 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CCBLPEN**: CCB clock enable in low-power mode
Set and cleared by software.
0: CCB peripheral clock disabled in low-power mode
1: CCB peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 4 **SAESLPEN**: SAES clock enable in low-power mode
Set and cleared by software.
0: SAES peripheral clock disabled in low-power mode
1: SAES peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x/55x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 3 **PKALPEN**: PKA clock enable in low-power mode
Set and cleared by software.
0: PKA peripheral clock disabled in low-power mode
1: PKA peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 2 **RNGLPEN**: RNG clock enable in low-power mode
Set and cleared by software.
0: RNG peripheral clock disabled in low-power mode
1: RNG peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 1 **HASHLPEN**: HASH clock enable in low-power mode
Set and cleared by software.
0: HASH peripheral clock disabled in low-power mode
1: HASH peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 0 **AESLPEN**: AES clock enable in low-power mode
Set and cleared by software.
0: AES peripheral clock disabled in low-power mode
1: AES peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/55x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

11.8.34 RCC APB1 peripheral low-power clock enable register (RCC_APB1LLPENR)

Address offset: 0x0C4

Reset value: 0x03FE DFF3

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	UART6LPEN	CRSLPEN	I3C1LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.
						rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3LPEN	SPI2LPEN	Res.	WWDGLPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM11LPEN	TIM10LPEN	TIM9LPEN	TIM7LPEN	TIM6LPEN	Res.	Res.	TIM3LPEN	TIM2LPEN
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **UART6LPEN**: UART6 clock enable in low-power mode
Set and cleared by software.
0: UART6 peripheral clock disabled in low-power mode
1: UART6 peripheral clock enabled in low-power mode (default after reset)

Bit 24 **CRSLPEN**: CRS clock enable in low-power mode
Set and cleared by software.
0: CRS peripheral clock disabled in low-power mode
1: CRS peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 23 **I3C1LPEN**: I3C1 clock enable in low-power mode
Set and cleared by software.
0: I3C1 peripheral clock disabled in low-power mode
1: I3C1 peripheral clock enabled in low-power mode (default after reset)

Bit 22 **I2C2LPEN**: I2C2 clock enable in low-power mode
Set and cleared by software.
0: I2C2 peripheral clock disabled in low-power mode
1: I2C2 peripheral clock enabled in low-power mode (default after reset)

Bit 21 **I2C1LPEN**: I2C1 clock enable in low-power mode
Set and cleared by software.
0: I2C1 peripheral clock disabled in low-power mode
1: I2C1 peripheral clock enabled in low-power mode (default after reset)

Bit 20 **UART5LPEN**: UART5 clock enable in low-power mode
Set and cleared by software.
0: UART5 peripheral clock disabled in low-power mode
1: UART5 peripheral clock enabled in low-power mode (default after reset)

Bit 19 **UART4LPEN**: UART4 clock enable in low-power mode
Set and cleared by software.
0: UART4 peripheral clock disabled in low-power mode
1: UART4 peripheral clock enabled in low-power mode (default after reset)

Bit 18 **USART3LPEN**: USART3 clock enable in low-power mode
Set and cleared by software.
0: USART3 peripheral clock disabled in low-power mode
1: USART3 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 17 **USART2LPEN**: USART2 clock enable in low-power mode
Set and cleared by software.
0: USART2 peripheral clock disabled in low-power mode
1: USART2 peripheral clock enabled in low-power mode (default after reset)

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3LPEN**: SPI3 clock enable in low-power mode
Set and cleared by software.
0: SPI3 peripheral clock disabled in low-power mode
1: SPI3 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **SPI2LPEN**: SPI2 clock enable in low-power mode
Set and cleared by software.
0: SPI2 peripheral clock disabled in low-power mode
1: SPI2 peripheral clock enabled in low-power mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **WWDGLPEN**: WWDG clock enable in low-power mode
Set and cleared by software. It is forced to one by hardware when the hardware WWDG option is activated.
0: WWDG peripheral clock disabled in low-power mode
1: WWDG peripheral clock enabled in low-power mode (default after reset)

Bit 11 **TIM14LPEN**: TIM14 clock enable in low-power mode
Set and cleared by software.
0: TIM14 peripheral clock disabled in low-power mode
1: TIM14 peripheral clock enabled in low-power mode (default after reset)

Bit 10 **TIM13LPEN**: TIM13 clock enable in low-power mode
Set and cleared by software.
0: TIM13 peripheral clock disabled in low-power mode
1: TIM13 peripheral clock enabled in low-power mode (default after reset)

Bit 9 **TIM12LPEN**: TIM12 clock enable in low-power mode
Set and cleared by software.
0: TIM12 peripheral clock disabled in low-power mode
1: TIM12 peripheral clock enabled in low-power mode (default after reset)

Bit 8 **TIM11LPEN**: TIM11 clock enable in low-power mode
Set and cleared by software.
0: TIM11 peripheral clock disabled in low-power mode
1: TIM11 peripheral clock enabled in low-power mode (default after reset)

Bit 7 **TIM10LPEN**: TIM10 clock enable in low-power mode
Set and cleared by software.
0: TIM10 peripheral clock disabled in low-power mode
1: TIM10 peripheral clock enabled in low-power mode (default after reset)

Bit 6 **TIM9LPEN**: TIM9 clock enable in low-power mode
Set and cleared by software.
0: TIM9 peripheral clock disabled in low-power mode
1: TIM9 peripheral clock enabled in low-power mode (default after reset)

Bit 5 **TIM7LPEN**: TIM7 clock enable in low-power mode
Set and cleared by software.
0: TIM7 peripheral clock disabled in low-power mode
1: TIM7 peripheral clock enabled in low-power mode (default after reset)

Bit 4 **TIM6LPEN**: TIM6 clock enable in low-power mode
Set and cleared by software.
0: TIM6 peripheral clock disabled in low-power mode
1: TIM6 peripheral clock enabled in low-power mode (default after reset)

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM3LPEN**: TIM3 clock enable in low-power mode
Set and cleared by software.
0: TIM3 peripheral clock disabled in low-power mode
1: TIM3 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 0 **TIM2LPEN**: TIM2 clock enable in low-power mode
Set and cleared by software.
0: TIM2 peripheral clock disabled in low-power mode
1: TIM2 peripheral clock enabled in low-power mode (default after reset)

11.8.35 RCC APB1 peripheral low-power clock enable register (RCC_APB1HLPENR)

Address offset: 0x0C8

Reset value: 0x4080 0208

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.
								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	Res.	Res.	Res.	DTSLPEN	Res.	Res.	Res.
						rw						rw			

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UCPD1LPEN**: UCPD1 clock enable in low-power mode
Set and cleared by software.
0: UCPD1 peripheral clock disabled in low-power mode
1: UCPD1 peripheral clock enabled in low-power mode (default after reset)

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FDCANLPEN**: FDCAN1, FDCAN2 and FDCAN3 peripheral clock enable in low-power mode
Set and cleared by software.
0: FDCAN1, FDCAN2 and FDCAN3 peripheral clock disabled in low-power mode
1: FDCAN1, FDCAN2 and FDCAN3 peripheral clock enabled in low-power mode (default after reset)

Note: Some STM32G56x/55x/54x/53x devices only have one FDCAN instance. Refer to the device datasheet for availability.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **DTSLPEN**: DTS clock enable in low-power mode
Set and cleared by software.
0: DTS peripheral clock disabled in low-power mode
1: DTS peripheral clock enabled in low-power mode (default after reset)

Bits 2:0 Reserved, must be kept at reset value.

11.8.36 RCC APB2 peripheral low-power clock enable register (RCC_APB2LPENR)

Address offset: 0x0CC

Reset value: 0x6107 7809

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TIM5LPEN	TIM4LPEN	Res.	Res.	Res.	Res.	USBLPEN	Res.	Res.	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN
	rw	rw					rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1LPEN	TIM8LPEN	SPI1LPEN	TIM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLAY1LPEN	Res.	Res.	HRTIM1LPEN
	rw	rw	rw	rw								rw			rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TIM5LPEN**: TIM5 clock enable in low-power mode
Set and cleared by software.
0: TIM5 peripheral clock disabled in low-power mode
1: TIM5 peripheral clock enabled in low-power mode (default after reset)

Bit 29 **TIM4LPEN**: TIM4 clock enable in low-power mode
Set and cleared by software.
0: TIM4 peripheral clock disabled in low-power mode
1: TIM4 peripheral clock enabled in low-power mode (default after reset)

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **USBLPEN**: USB clock enable in low-power mode
Set and cleared by software.
0: USB peripheral clock disabled in low-power mode
1: USB peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TIM17LPEN**: TIM17 clock enable in low-power mode
Set and cleared by software.
0: TIM17 peripheral clock disabled in low-power mode
1: TIM17 peripheral clock enabled in low-power mode (default after reset)

Bit 17 **TIM16LPEN**: TIM16 clock enable in low-power mode
Set and cleared by software.
0: TIM16 peripheral clock disabled in low-power mode
1: TIM16 peripheral clock enabled in low-power mode (default after reset)

Bit 16 **TIM15LPEN**: TIM15 clock enable in low-power mode
Set and cleared by software.
0: TIM15 peripheral clock disabled in low-power mode
1: TIM15 peripheral clock enabled in low-power mode (default after reset)

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1LPEN**: USART1 clock enable in low-power mode
Set and cleared by software.
0: USART1 peripheral clock disabled in low-power mode
1: USART1 peripheral clock enabled in low-power mode (default after reset)

Bit 13 **TIM8LPEN**: TIM8 clock enable in low-power mode
Set and cleared by software.
0: TIM8 peripheral clock disabled in low-power mode
1: TIM8 peripheral clock enabled in low-power mode (default after reset)

Bit 12 **SPI1LPEN**: SPI1 clock enable in low-power mode
Set and cleared by software.
0: SPI1 peripheral clock disabled in low-power mode
1: SPI1 peripheral clock enabled in low-power mode (default after reset)

Bit 11 **TIM1LPEN**: TIM1 clock enable in low-power mode
Set and cleared by software.
0: TIM1 peripheral clock disabled in low-power mode
1: TIM1 peripheral clock enabled in low-power mode (default after reset)

Bits 10:4 Reserved, must be kept at reset value.

Bit 3 **PLAY1LPEN**: PLAY1 clock enable in low-power mode
Set and cleared by software.
0: PLAY1 peripheral clock disabled in low-power mode
1: PLAY1 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **HRTIM1LPEN**: HRTIM1 clock enable in low-power mode
Set and cleared by software.
0: HRTIM1 peripheral clock disabled in low-power mode
1: HRTIM1 peripheral clock enabled in low-power mode (default after reset)

11.8.37 RCC APB3 peripheral low-power clock enable register (RCC_APB3LPENR)

Address offset: 0x0D0

Reset value: 0xFFC7 C9DE

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP8LPEN	COMP7LPEN	COMP6LPEN	COMP5LPEN	COMP4LPEN	COMP3LPEN	COMP2LPEN	COMP1LPEN	RTCAPBLPEN	VREFLPEN	Res.	Res.	Res.	OPAMP3LPEN	OPAMP2LPEN	OPAMP1LPEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POSENC2LPEN	POSENC1LPEN	Res.	Res.	LPTIM1LPEN	Res.	Res.	I2C4LPEN	I2C3LPEN	LPUART1LPEN	Res.	TIM27LPEN	TIM26LPEN	TIM20LPEN	SBSLPEN	Res.
rw	rw			rw			rw	rw	rw		rw	rw	rw	rw	

Bit 31 **COMP8LPEN**: COMP8 clock enable in low-power mode
Set and cleared by software.
0: COMP8 peripheral clock disabled in low-power mode
1: COMP8 peripheral clock enabled in low-power mode (default after reset)

Bit 30 **COMP7LPEN**: COMP7 clock enable in low-power mode
Set and cleared by software.
0: COMP7 peripheral clock disabled in low-power mode
1: COMP7 peripheral clock enabled in low-power mode (default after reset)

Bit 29 **COMP6LPEN**: COMP6 clock enable in low-power mode
Set and cleared by software.
0: COMP6 peripheral clock disabled in low-power mode
1: COMP6 peripheral clock enabled in low-power mode (default after reset)

Bit 28 **COMP5LPEN**: COMP5 clock enable in low-power mode
Set and cleared by software.
0: COMP5 peripheral clock disabled in low-power mode
1: COMP5 peripheral clock enabled in low-power mode (default after reset)

Bit 27 **COMP4LPEN**: COMP4 clock enable in low-power mode
Set and cleared by software.
0: COMP4 peripheral clock disabled in low-power mode
1: COMP4 peripheral clock enabled in low-power mode (default after reset)

Bit 26 **COMP3LPEN**: COMP3 clock enable in low-power mode
Set and cleared by software.
0: COMP3 peripheral clock disabled in low-power mode
1: COMP3 peripheral clock enabled in low-power mode (default after reset)

Bit 25 **COMP2LPEN**: COMP2 clock enable in low-power mode
Set and cleared by software.
0: COMP2 peripheral clock disabled in low-power mode
1: COMP2 peripheral clock enabled in low-power mode (default after reset)

Bit 24 **COMP1LPEN**: COMP1 clock enable in low-power mode
Set and cleared by software.
0: COMP1 peripheral clock disabled in low-power mode
1: COMP1 peripheral clock enabled in low-power mode (default after reset)

Bit 23 **RTCAPBLPEN**: RTC APB interface clock enable in low-power mode
Set and cleared by software.
0: RTC APB interface clock disabled in low-power mode
1: RTC APB interface clock enabled in low-power mode (default after reset)

Bit 22 **VREFLPEN**: VREFBUF clock enable in low-power mode
Set and cleared by software.
0: VREFBUF peripheral clock disabled in low-power mode
1: VREFBUF peripheral clock enabled in low-power mode (default after reset)

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **OPAMP3LPEN**: OPAMP3 clock enable in low-power mode
Set and cleared by software.
0: OPAMP3 peripheral clock disabled in low-power mode
1: OPAMP3 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 17 **OPAMP2LPEN**: OPAMP2 clock enable in low-power mode
Set and cleared by software.
0: OPAMP2 peripheral clock disabled in low-power mode
1: OPAMP2 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 16 **OPAMP1LPEN**: OPAMP1 clock enable in low-power mode
Set and cleared by software.
0: OPAMP1 peripheral clock disabled in low-power mode
1: OPAMP1 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 15 **POSENC2LPEN**: POSENC2 clock enable in low-power mode
Set and cleared by software.
0: POSENC2 peripheral clock disabled in low-power mode
1: POSENC2 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 14 **POSENC1LPEN**: POSENC1 clock enable in low-power mode
Set and cleared by software.
0: POSENC1 peripheral clock disabled in low-power mode
1: POSENC1 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **LPTIM1LPEN**: LPTIM1 clock enable in low-power mode
Set and cleared by software.
0: LPTIM1 peripheral clock disabled in low-power mode
1: LPTIM1 peripheral clock enabled in low-power mode (default after reset)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **I2C4LPEN**: I2C4 clock enable in low-power mode
Set and cleared by software.
0: I2C4 peripheral clock disabled in low-power mode
1: I2C4 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 7 **I2C3LPEN**: I2C3 clock enable in low-power mode
Set and cleared by software.
0: I2C3 peripheral clock disabled in low-power mode
1: I2C3 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 6 **LPUART1LPEN**: LPUART1 clock enable in low-power mode
Set and cleared by software.
0: LPUART1 peripheral clock disabled in low-power mode
1: LPUART1 peripheral clock enabled in low-power mode (default after reset)

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TIM27LPEN**: TIM27 clock enable in low-power mode
Set and cleared by software.
0: TIM27 peripheral clock disabled in low-power mode
1: TIM27 peripheral clock enabled in low-power mode (default after reset)

Bit 3 **TIM26LPEN**: TIM26 clock enable in low-power mode
Set and cleared by software.
0: TIM26 peripheral clock disabled in low-power mode
1: TIM26 peripheral clock enabled in low-power mode (default after reset)

Bit 2 **TIM20LPEN**: TIM20 clock enable in low-power mode
Set and cleared by software.
0: TIM20 peripheral clock disabled in low-power mode
1: TIM20 peripheral clock enabled in low-power mode (default after reset)
Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 1 **SBSLPEN**: SBS clock enable in low-power mode
Set and cleared by software.
0: SBS peripheral clock disabled in low-power mode
1: SBS peripheral clock enabled in low-power mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

11.8.38 RCC kernel clock configuration register (RCC_CCIPR1)

Address offset: 0x0D8

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALCKEN	Res.	Res.	Res.	Res.	LPUART1SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	UART6SEL[2:1]	
rw					rw	rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART6SEL[0]	UART5SEL[2:0]			UART4SEL[2:0]			USART3SEL[2:0]			USART2SEL[2:0]			USART1SEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CALCKEN**: Calibration clock enable
Set and cleared by software.
0: Calibration clock (hsi_cal_ck) disabled (default after reset)
1: Calibration clock (hsi_cal_ck) enabled

Bits 30:27 Reserved, must be kept at reset value.

Bits 26:24 **LPUART1SEL[2:0]**: LPUART1 kernel clock source selection
Set and cleared by software.
000: rcc_pclk3 s elected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_q_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:15 **UART6SEL[2:0]**: UART6 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_q_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bits 14:12 **UART5SEL[2:0]**: UART5 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_q_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bits 11:9 **UART4SEL[2:0]**: UART4 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_q_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bits 8:6 **USART3SEL[2:0]**: USART3 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_q_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 5:3 **USART2SEL[2:0]**: USART2 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_q_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bits 2:0 **USART1SEL[2:0]**: USART1 kernel clock source selection
Set and cleared by software.
000: rcc_pclk2 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_q_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

1. This source is not available on STM32G53x/54x devices. When the source is not available, the corresponding input must not be selected. Refer to the device datasheet for availability

11.8.39 RCC kernel clock configuration register (RCC_CCIPR2)

Address offset: 0x0DC

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DAC10SEL[2:0]			Res.	DAC9SEL[2:0]			Res.	DAC8SEL[2:0]			Res.	DAC7SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DAC6SEL[2:0]			Res.	DAC5SEL[2:0]			DAC2SHSEL	DAC2SEL[2:0]			DAC1SHSEL	DAC1SEL[2:0]		
	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DAC10SEL[2:0]**: DAC10 kernel clock source selection
Set and cleared by software.
000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)
001: rcc_hrtim_ker_ck selected as kernel clock
010: rcc_adc_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: pll1_q_ck selected as kernel clock
110: pll2_q_ck selected as kernel clock (1)
111: reserved, the kernel clock is disabled

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DAC9SEL[2:0]**: DAC9 kernel clock source selection
Set and cleared by software.
000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)
001: rcc_hrtim_ker_ck selected as kernel clock
010: rcc_adc_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: pll1_q_ck selected as kernel clock
110: pll2_q_ck selected as kernel clock (1)
111: reserved, the kernel clock is disabled

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DAC8SEL[2:0]**: DAC8 kernel clock source selection
Set and cleared by software.
000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)
001: rcc_hrtim_ker_ck selected as kernel clock
010: rcc_adc_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: pll1_q_ck selected as kernel clock
110: pll2_q_ck selected as kernel clock (1)
111: reserved, the kernel clock is disabled

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DAC7SEL[2:0]**: DAC7 kernel clock source selection
Set and cleared by software.
000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)
001: rcc_hrtim_ker_ck selected as kernel clock
010: rcc_adc_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: pll1_q_ck selected as kernel clock
110: pll2_q_ck selected as kernel clock (1)
111: reserved, the kernel clock is disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DAC6SEL[2:0]**: DAC6 kernel clock source selection
Set and cleared by software.
000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)
001: rcc_hrtim_ker_ck selected as kernel clock
010: rcc_adc_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: pll1_q_ck selected as kernel clock
110: pll2_q_ck selected as kernel clock (1)
111: reserved, the kernel clock is disabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DAC5SEL[2:0]**: DAC5 kernel clock source selection
Set and cleared by software.
000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)
001: rcc_hrtim_ker_ck selected as kernel clock
010: rcc_adc_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: pll1_q_ck selected as kernel clock
110: pll2_q_ck selected as kernel clock (1)
111: reserved, the kernel clock is disabled

Bit 7 **DAC2SHSEL**: DAC2 sample and hold clock
Set and cleared by software.
0: lse_ck selected as kernel clock (default after reset)
1: lsi_ck selected as kernel clock

Bits 6:4 **DAC2SEL[2:0]**: DAC2 kernel clock source selection
Set and cleared by software.
000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)
001: rcc_hrtim_ker_ck selected as kernel clock
010: rcc_adc_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: pll1_q_ck selected as kernel clock
110: pll2_q_ck selected as kernel clock (1)
111: reserved, the kernel clock is disabled

Bit 3 **DAC1SHSEL**: DAC1 sample and hold clock
Set and cleared by software.
0: lse_ck selected as kernel clock (default after reset)
1: lsi_ck selected as kernel clock

Bits 2:0 **DAC1SEL[2:0]**: DAC1 kernel clock source selection
Set and cleared by software.
000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)
001: rcc_hrtim_ker_ck selected as kernel clock
010: rcc_adc_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: pll1_q_ck selected as kernel clock
110: pll2_q_ck selected as kernel clock (1)
111: reserved, the kernel clock is disabled

1. This source is not available on STM32G53x/54x devices. When the source is not available, the corresponding input must not be selected. Refer to the device datasheet for availability

11.8.40 RCC kernel clock configuration register (RCC_CCIPR3)

Address offset: 0x0E0

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SYSTICKSEL[1:0]		Res.	LPTIM1SEL[2:0]			PLAY1DIV[3:0]				Res.	PLAY1SEL[2:0]		
		rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI3SEL[2:0]			SPI2SEL[2:0]			SPI1SEL[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **SYSTICKSEL[1:0]**: SYSTICK clock source selection

Set and cleared by software.

00: rcc_hclk/8 selected as clock source (default after reset)

01: lsi_ck selected as clock source

10: lse_ck selected as clock source

11: reserved, the kernel clock is disabled

*Note: rcc_hclk frequency must be four times higher than lsi_ck/lse_ck, that is, period (LSI/LSE) / 4 * period (HCLK).*

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **LPTIM1SEL[2:0]**: LPTIM1 kernel clock source selection

Set and cleared by software.

000: rcc_pclk3 selected as kernel clock (default after reset)

001: pll1_q_ck selected as kernel clock

010: pll2_p_ck selected as kernel clock (1)

011: hse_ker_ck selected as kernel clock

100: hsi_ker_ck selected as kernel clock

101: lsi_ck selected as kernel clock

110: lse_ck selected as kernel clock

111: reserved, the kernel clock is disabled

Bits 23:20 **PLAY1DIV[3:0]**: PLAY1 kernel clock divider
Set and cleared by software
0000: Selected kernel is not divided (default after reset)
0001: Selected kernel clock is divided by 2
0010: Selected kernel clock is divided by 4
0011: Selected kernel clock is divided by 8
0100: Selected kernel clock is divided by 16
0101: Selected kernel clock is divided by 32
0110: Selected kernel clock is divided by 64
0111: Selected kernel clock is divided by 128
1000: Selected kernel clock is divided by 256
1001: Selected kernel clock is divided by 512
1010: Selected kernel clock is divided by 1024
1011: Selected kernel clock is divided by 2048
1100: Selected kernel clock is divided by 4096
1101: Selected kernel clock is divided by 8192
1110: Selected kernel clock is divided by 16384
1111: Selected kernel clock is divided by 32768
This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PLAY1SEL[2:0]**: PLAY1 kernel clock source selection
Set and cleared by software.
000: lse_ck selected as kernel clock (default after reset)
001: lsi_ck selected as kernel clock
010: pll2_p_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: hsi48_ker_ck selected as kernel clock (1)
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:6 **SPI3SEL[2:0]**: SPI3 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_p_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: reserved, the kernel clock is disabled
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 5:3 **SPI2SEL[2:0]**: SPI2 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_p_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: AUDIOCLK selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bits 2:0 **SPI1SEL[2:0]**: SPI1 kernel clock source selection
Set and cleared by software.
000: rcc_pclk2 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: pll2_p_ck selected as kernel clock (1)
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: AUDIOCLK selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

1. This source is not available on STM32G53x/54x devices. When the source is not available, the corresponding input must not be selected. Refer to the device datasheet for availability.

11.8.41 RCC kernel clock configuration register (RCC_CCIPR4)

Address offset: 0x0E4

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	I2C4SEL[2:0]			Res.	I2C3SEL[2:0]			Res.	I2C2SEL[2:0]			Res.	I2C1SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	I3C1SEL[2:0]			Res.	Res.	USBSEL[1:0]		Res.	Res.	UCPD1SEL[1:0]		Res.	XSPI1SEL[2:0]		
	rw	rw	rw			rw	rw			rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **I2C4SEL[2:0]**: I2C4 kernel clock source selection
Set and cleared by software.
000: rcc_pclk3 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: hsi_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi48_ker_ck selected as kernel clock
101: reserved, the kernel clock is disabled
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **I2C3SEL[2:0]**: I2C3 kernel clock source selection
Set and cleared by software.
000: rcc_pclk3 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: hsi_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi48_ker_ck selected as kernel clock
101: reserved, the kernel clock is disabled
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **I2C2SEL[2:0]**: I2C2 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: hsi_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: reserved, the kernel clock is disabled
101: reserved, the kernel clock is disabled
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **I2C1SEL[2:0]**: I2C1 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: hsi_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: reserved, the kernel clock is disabled
101: reserved, the kernel clock is disabled
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **I3C1SEL[2:0]**: I3C1 kernel clock source selection
Set and cleared by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: ic1_ck selected as kernel clock
010: hsi_ker_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: reserved, the kernel clock is disabled
101: reserved, the kernel clock is disabled
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **USBSEL[1:0]**: USB kernel clock source selection
Set and cleared by software.
00: hsi48_ker_ck selected as kernel clock (default after reset)
01: hse_ker_ck selected as kernel clock
10: pll1_q_ck selected as kernel clock
11: pll2_q_ck selected as kernel clock

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **UCPD1SEL[1:0]**: UCPD1 clock source selection
Set and cleared by software.
00: hsi_ker_ck selected as kernel clock (default after reset)
01: hse_ker_ck selected as kernel clock
10: hsi48_ker_ck selected as kernel clock
11: pll2_q_ck selected as kernel clock

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **XSPI1SEL[2:0]**: XSPI1 kernel clock source selection
Set and cleared by software.
000: rcc_hclk4 selected as kernel clock (default after reset)
001: pll1_q_ck selected as kernel clock
010: ic2_ck selected as kernel clock
011: hse_ker_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: reserved, the kernel clock is disabled
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

11.8.42 RCC kernel clock configuration register (RCC_CCIPR5)

Address offset: 0x0E8

Reset value: 0x0000 0000

This register can be accessed in secure or nonsecure mode, depending on the security level of each peripheral. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSENC2SEL[2:0]			Res.	POSENC1SEL[2:0]		
									rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FDCANSEL[1:0]		Res.	Res.	Res.	Res.	ADCDIV[3:0]				Res.	ADCSEL[2:0]		
		rw	rw					rw	rw	rw	rw		rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **POSENC2SEL[2:0]**: POSENC2 kernel clock source selection
Set and cleared by software.
000: pll1_p_ck selected as kernel clock (default after reset)
001: pll1_q_ck selected as kernel clock
010: ic1_ck selected as kernel clock
011: pll2_p_ck selected as kernel clock
100: pll2_q_ck selected as kernel clock
101: ic2_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **POSENC1SEL[2:0]**: POSENC1 kernel clock source selection
Set and cleared by software.
000: pll1_p_ck selected as kernel clock (default after reset)
001: pll1_q_ck selected as kernel clock
010: ic1_ck selected as kernel clock
011: pll2_p_ck selected as kernel clock
100: pll2_q_ck selected as kernel clock
101: ic2_ck selected as kernel clock
110: reserved, the kernel clock is disabled
111: reserved, the kernel clock is disabled

Note: This bitfield is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **FDCANSEL[1:0]**: FDCAN1, FDCAN2, and FDCAN3 kernel clock source selection
Set and cleared by software.
00: hse_ker_ck selected as kernel clock (default after reset)
01: pll1_q_ck selected as kernel clock
10: pll2_q_ck selected as kernel clock (1)
11: reserved, the kernel clock is disabled

Note: Some STM32G56x/55x/54x/53x devices only have one FDCAN instance. Refer to the device datasheet for availability.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **ADCDIV[3:0]**: ADC clock prescaler
Set and cleared by software.
This bitfields selects the divider value on the ADC kernel clock.
0000: ADC kernel clock not divided (default after reset)
0001: ADC kernel clock divided by 2
0010: ADC kernel clock divided by 4
0011: ADC kernel clock divided by 6
0100: ADC kernel clock divided by 8
0101: ADC kernel clock divided by 10
0110: ADC kernel clock divided by 12
0111: ADC kernel clock divided by 16
1000: ADC kernel clock divided by 32
1001: ADC kernel clock divided by 64
1010: ADC kernel clock divided by 128
1011: ADC kernel clock divided by 256
others: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **ADCSEL[2:0]**: ADC12 and ADC3 kernel clock source selection

Set and cleared by software.

000: bus_ck (same as rcc_tim_ker_ck) selected as kernel clock (default after reset)

001: rcc_hrtim_ker_ck selected as kernel clock

010: hsi_ker_ck selected as kernel clock

011: hse_ker_ck selected as kernel clock

100: pll1_q_ck selected as kernel clock

101: ic1_ck selected as kernel clock

110: pll2_p_ck selected as kernel clock (1)

111: ic2_ck selected as kernel clock (1)

Note: ADC3 is not available on STM32G53x/54x devices. Refer to the device datasheet for availability.

1. This source is not available on STM32G53x/54x devices. When the source is not available, the corresponding input must not be selected. Refer to the device datasheet for availability.

11.8.43 RCC RTC domain control register (RCC_RTCCR)

Address offset: 0x0F0

Reset value: 0x0000 0000

Reset by RTC domain reset.

Access time: between 0 and 3 wait states, size can be word, half-word half-word, and byte access. Wait states are inserted in case of successive accesses to this register.

Note: After a system reset, this register is write-protected by DRTCP (except LSION and RTCEN bits). To modify these bits, the DRTCP bit of the PWR_RTCCR *must be set. All bits are reset only after an RTC domain reset (see Section 11.4.3).*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSIRDY	LSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCDRST
				r	rw										rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw						rwo	rwo		r	rw	rw	rw	rw	r	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **LSIRDY**: LSI oscillator ready

Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after three internal low-speed oscillator clock cycles.

This bit is set when the LSI is used by IWDG or RTC, even if LSION = 0.

0: LSI oscillator not ready (default after power-on reset)

1: LSI oscillator ready

Bit 26 **LSION**: LSI oscillator enable
Set and cleared by software.
0: LSI oscillator off (default after power-on reset)
1: LSI oscillator on

Bits 25:17 Reserved, must be kept at reset value.

Bit 16 **RTCDRST**: RTC domain software reset
Set and cleared by software.
0: reset not activated (default after RTC domain reset)
1: resets the entire RTC domain

Bit 15 **RTCEN**: RTC clock enable
Set and cleared by software.
0: rtc_ck disabled (default after RTC domain reset)
1: rtc_ck enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**:RTC clock source selection
Set by software to select the clock source for the RTC.
These bits can be written only once, or when LSECSSD = 0.
The RTCDRST bit can be used to reset them.
If HSE is selected as RTC clock, this clock is lost when the system is in Stop mode or in case of a pin reset (NRST).
00: no clock (default after RTC domain reset)
01: LSE selected as RTC clock
10: LSI selected as RTC clock
11: HSE divided by RTCPRE value selected as RTC clock

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LSECSSD**: LSE clock security system failure detection
Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator. This bit is reset when LSECSSON is cleared.
0: no failure detected on 32 kHz oscillator (default after V_{DD} domain reset)
1: failure detected on 32 kHz oscillator

Bit 5 **LSECSSON**: LSE clock security system enable
Set by software to enable the clock security system on LSE oscillator.
LSECSSON must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware) and after RTCSEL is selected.
0: CSS on LSE oscillator OFF (default after V_{DD} domain reset)
1: CSS on LSE oscillator ON

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator driving capability
Set by software to select the driving capability of the LSE oscillator.
This field can be written only if LSE oscillator is disabled (LSEON = 0). The user must also wait for LSERDY = 0.
0x: medium-low drive (default after V_{DD} domain reset)
10: medium-high drive
11: highest drive

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass oscillator in debug mode. This bit must not be written when the LSE is enabled (by LSEON) or ready (LSERDY = 1)

0: LSE oscillator not bypassed (default after V_{DD} domain reset)

1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready

Set and cleared by hardware to indicate when the LSE is stable.

This bit needs six cycles of lse_ck clock to fall down after LSEON has been cleared.

0: LSE oscillator not ready (default after V_{DD} domain reset)

1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enabled

Set and cleared by software.

0: LSE oscillator OFF (default after VDD domain reset)

1: LSE oscillator ON

11.8.44 RCC reset status register (RCC_RSR)

Address offset: 0x0F4

Reset value: 0x0C00 0000

Reset by power-on reset only.

Access: between 0 half-word, and 3 wait states, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

RMVF bit can be accessed in secure or nonsecure mode, depending on RMVSEC bit. The privileged access depends on the values of the SPRIV and NSPRIV bits of the *RCC privilege configuration register (RCC_PRIVCFGR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r			w							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **LPWRRSTF**: Low-power reset flag

Set by hardware when a reset occurs due to Stop or Standby mode entry, whereas the corresponding nRST_STOP, nRST_STBY option bit is cleared.
Cleared by writing to the RMVF bit.
0: No illegal low-power mode reset occurred
1: Illegal low-power mode reset occurred

Bit 30 **WWDGRSTF**: window watchdog reset flag

Cleared by software by writing the RMVF bit.
Set by hardware when a window watchdog reset occurs.
0: no window watchdog reset occurred from WWDG (default after power-on reset)
1: window watchdog reset occurred from WWDG

Bit 29 **IWDGRSTF**: independent watchdog reset flag

Cleared by software by writing the RMVF bit.
Set by hardware when an independent watchdog reset occurs.
0: no independent watchdog reset occurred (default after power-on reset)
1: independent watchdog reset occurred

Bit 28 **SFTRSTF**: system reset from CPU reset flag

Cleared by software by writing the RMVF bit.
Set by hardware when the system reset is due to CPU.The CPU can generate a system reset by writing SYSRESETREQ bit of the AIRCR register of the Cortex® M33.
0: no CPU software reset occurred (default after power-on reset)
1: a system reset has been generated by the CPU

Bit 27 **PORRSTF**: POR reset flag

Cleared by software by writing the RMVF bit.
Set by hardware when a POR reset occurs (pwr_por_rst).
0: no POR reset occurred
1: POR reset occurred (default after power-on reset)

Bit 26 **PINRSTF**: pin reset flag (NRST)
Cleared by software by writing the RMVF bit.
Set by hardware when a reset from pin occurs.
0: no reset from pin occurred
1: reset from pin occurred (default after power-on reset)

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: remove reset flag
Set by software and reset by hardware.
0: reset of the reset flags not activated (default after power-on reset)
1: resets the value of the reset flags

Bits 22:0 Reserved, must be kept at reset value.

11.8.45 RCC secure configuration register (RCC_SECCFGR)

Address offset: 0x110

Reset value: 0x0000 0000

Access: no wait state; word, half-word half-word, and byte access

When the system is secure (TZEN = 0xB4), this register can be written only by a secure privileged access if SPRIV = 1, and by a secure privileged or unprivileged access if SPRIV = 0. A nonsecure write access generates an illegal access event and data is not written. This register can be read by secure or nonsecure, privilege or unprivileged access. When the system is not secure (TZEN = 0xC3), this register is read as 0 and the register write is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RMVFSEC	HSI48SEC	Res.	Res.	PLL2SEC	PLL1SEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	Res.	HSESEC	HSISEC
			rw	rw			rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RMVFSEC**: Remove reset flag security
Set and cleared by software.
0: nonsecure
1: secure

Bit 11 **HSI48SEC**: HSI48 clock configuration and status bits security
Set and cleared by software.
0: nonsecure
1: secure

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **PLL2SEC**: PLL2 clock configuration and status bits security
Set and cleared by software.
0: nonsecure
1: secure

Note: This bit is not available on STM32G53x/54x devices. If not present, it must be kept to its reset value. Refer to the device datasheet for availability.

Bit 7 **PLL1SEC**: PLL1 clock configuration and status bits security
Set and cleared by software.
0: nonsecure
1: secure

Bit 6 **PRESCSEC**: AHBx/APBx prescaler configuration bits security
Set and cleared by software.
0: nonsecure
1: secure

Bit 5 **SYSCLKSEC**: SYSCLK clock selection, clock output on MCO configuration security
Set and cleared by software.
0: nonsecure
1: secure

Bit 4 **LSESEC**: LSE clock configuration and status bits security
Set and cleared by software.
0: nonsecure
1: secure

Bit 3 **LSISEC**: LSI clock configuration and status bits security
Set and cleared by software.
0: nonsecure
1: secure

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HSESEC**: HSE clock configuration bits, status bits and HSE_CSS security
Set and cleared by software.
0: nonsecure
1: secure

Bit 0 **HSISEC**: HSI clock configuration and status bits security
Set and cleared by software.
0: nonsecure
1: secure

11.8.46 RCC privilege configuration register (RCC_PRIVCFGR)

Address offset: 0x114

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register can be written only by a privileged access. It can be read by privileged or unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: RCC nonsecure functions privilege configuration

Set and cleared by software. This bit can be written only by privileged access, secure or nonsecure.

0: Read and write to RCC nonsecure functions can be done by privileged or unprivileged access.

1: Read and write to RCC nonsecure functions can be done by privileged access only

Bit 0 **SPRIV**: RCC secure functions privilege configuration

Set and cleared by software. This bit can be written only by a secure privileged access.

0: Read and write to RCC secure functions can be done by privileged or unprivileged access.

1: Read and write to RCC secure functions can be done by privileged access only

11.8.47 RCC register map

Table 132. RCC register map and reset values

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Res.	Res.	Res.	Res.	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON	Res.	Res.	HSI48RDY	HSI48ON	Res.	Res.	Res.	Res.	Res.	Res.	HSIDIVF	HSIDIV[1:0]		HSIKERON	HSIRDY	HSION
	Reset value					0	0	0	0				0	0	0	0	0			0	0							1	0	0	0	1	1
0x004 - 0x00C	Reserved	Res.																															
0x010	RCC_HSICFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[6:0]							Res.	Res.	Res.	Res.	HSICAL[11:0]											
	Reset value										1	0	0	0	0	0	0					-	-	-	-	-	-	-	-	-	-	-	-
0x014	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
	Reset value																							-	-	-	-	-	-	-	-	-	-

Table 132. RCC register map and reset values (continued)

Offset	Register name / Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	Reserved	Res.																															
0x01C	RCC_CFGR1	MCO2SEL[2:0]			MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]				Res.	Res.	Res.	Res.	RTCPRE[5:0]						Res.	Res.	Res.	SYSSWS[1:0]		Res.	SYSSW[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0				0	0		0	0
0x020	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIS	APB2DIS	APB1DIS	AHB4DIS	AHB3DIS	AHB2DIS	AHB1DIS	Res.		PPRE3[2:0]		Res.	PPRE2[2:0]			Res.	PPRE1[2:0]		HPRE[3:0]				
	Reset value										0	0	0	0	0	0	0		0	0	0		0	0	0		0	0	0	0	0	0	0	0
0x028	RCC_PLL1CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ICLK1DIV	Res.	Res.	Res.	Res.	Res.	PLL1REN	PLL1QEN	PLL1PEN	Res.	Res.	PLL1M[5:0]						Res.	Res.	PLL1VCOSEL	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
	Reset value								0						0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x02C	RCC_PLL2CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ICLK2DIV	Res.	Res.	Res.	Res.	Res.	PLL2REN	PLL2QEN	PLL2PEN	Res.	Res.	PLL2M[5:0]						Res.	Res.	PLL2VCOSEL	PLL2FRACEN	PLL2RGE[1:0]		PLL2SRC[1:0]	
	Reset value								0						0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	RCC_PLL1DIVR	Res.	PLL1R[6:0]							Res.	PLL1Q[6:0]							PLL1P[6:0]								PLL1N[8:0]							
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x038	RCC_PLL1FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1FRACN[12:0]														Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x03C	RCC_PLL2DIVR	Res.	PLL2R[6:0]							Res.	PLL2Q[6:0]							PLL2P[6:0]								PLL2N[8:0]							
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x040	RCC_PLL2FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2FRACN[12:0]														Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x044 - 0x04C	Reserved	Res.																															
0x050	RCC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		LSECSSIE	Res.	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSIRDYIE
	Reset value																								0		0	0	0	0	0		0	0
0x054	RCC_IFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSF	LSECSSF	Res.	PLL2RDYF	PLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	Res.	LSERDYF	LSIRDYF
	Reset value																							0	0		0	0	0	0	0		0	0

Table 132. RCC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	RCC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSC	LSECSSC	Res.	PLL2RDYC	PLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	Res.	LSERDYC	LSIRDYC
	Reset value																						0	0		0	0	0	0	0		0	0
0x05C	Reserved	Res.																															
0x060	RCC_AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGRST	Res.	FMAC1RST	CORDIC1RST	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2RST	GPDMA1RST
	Reset value															0		0	0		0											0	0
0x064	RCC_AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3RST	ADC12RST	FMAC2RST	Res.	Res.	Res.	Res.	Res.	DAC10RST	DAC9RST	DAC8RST	DAC7RST	DAC6RST	DAC5RST	Res.	Res.	DAC2RST	DAC1RST
	Reset value															0	0	0						0	0	0	0	0	0			0	0
0x068	Reserved	Res.																															
0x06C	RCC_AHB4RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1RST	Res.	Res.	Res.	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST	Res.	OTFDEC1RST	Res.	CCBRST	SAESRST	PKARST	RNGRST	HASHRST	AESRST
	Reset value												0				0	0	0	0	0	0	0	0		0		0	0	0	0	0	0
0x070	Reserved	Res.																															
0x074	RCC_APB1LRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART6RST	CRSRST	I3C1RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	Res.	TIM14RST	TIM13RST	TIM12RST	TIM11RST	TIM10RST	TIM9RST	TIM7RST	TIM6RST	Res.	TIM3RST	TIM2RST
	Reset value							0	0	0	0	0	0	0	0	0		0	0			0	0	0	0	0	0	0	0		0	0
0x078	RCC_APB1HRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRS	Res.	Res.	Res.	Res.	Res.	DTSRST	Res.	Res.	Res.
	Reset value									0														0						0			
0x07C	RCC_APB2RSTR	Res.	TIM5RST	TIM4RST	Res.	Res.	Res.	Res.	USBRST	Res.	Res.	Res.	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	Res.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	I2C4RST	I2C3RST	LPUART1RST	Res.	TIM27RST	TIM26RST	TIM20RST	Res.	Res.
	Reset value		0	0					0							0	0	0		0	0	0	0							0	0	0		0
0x080	RCC_APB3RSTR	COMP8RST	COMP7RST	COMP6RST	COMP5RST	COMP4RST	COMP3RST	COMP2RST	COMP1RST	Res.	VREFRST	Res.	Res.	Res.	Res.	OPAMP3RST	OPAMP2RST	OPAMP1RST	POSENC2RST	POSENC1RST	Res.	Res.	LPTIM1RST	Res.	Res.	I2C4RST	I2C3RST	LPUART1RST	Res.	TIM27RST	TIM26RST	TIM20RST	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0		0					0	0	0	0	0			0			0	0	0		0	0	0		
0x084	Reserved	Res.																															

Table 132. RCC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	RCC_AHB1ENR	SRAM1EN	DCACHEEN	Res.	Res.	SRAM3EN	SRAM2EN	Res.	TZSC1EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	Res.	FMAC1EN	CORDIC1EN	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2EN	GPDMA1EN
	Reset value	1	1			1	1		0							0		0	0		0				1							0	0
0x08C	RCC_AHB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3EN	ADC12EN	FMAC2EN	Res.	Res.	Res.	Res.	Res.	DAC10EN	DAC9EN	DAC8EN	DAC7EN	DAC6EN	DAC5EN	Res.	Res.	DAC2EN	DAC1EN
	Reset value															0	0	0						0	0	0	0	0	0			0	0
0x090	RCC_AHB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCKEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0																					
0x094	RCC_AHB4ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1EN	Res.	Res.	Res.	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN	Res.	OTFDEC1EN	Res.	CCBEN	SAESEN	PKAEN	RNGEN	HASHEN	AESEN
	Reset value												0				0	0	0	0	0	0	0	0		0		0	0	0	0	0	0
0x098	Reserved	Res.																															
0x09C	RCC_APB1LENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART6EN	CRSEN	I3C1EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	WWDGEN	TIM14EN	TIM13EN	TIM12EN	TIM11EN	TIM10EN	TIM9EN	TIM7EN	TIM6EN	Res.	Res.	TIM3EN	TIM2EN
	Reset value								0	0	0	0	0	0	0	0		0	0		0	0	0	0	0	0	0	0	0			0	0
0x0A0	RCC_APB1HENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	Res.	Res.	Res.	DTSEN	Res.	Res.	Res.
	Reset value									0														0						0			
0x0A4	RCC_APB2ENR	Res.	TIM5EN	TIM4EN	Res.	Res.	Res.	Res.	USBEN	Res.	Res.	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	PLAY1EN	PLAY1APBEN	Res.	Res.	HRTIM1EN
	Reset value		0	0					0						0	0	0		0	0	0	0							0	0			0
0x0A8	RCC_APB3ENR	COMP8EN	COMP7EN	COMP6EN	COMP5EN	COMP4EN	COMP3EN	COMP2EN	COMP1EN	RTCAPBEN	VREFEN	Res.	Res.	Res.	OPAMP3EN	OPAMP2EN	OPAMP1EN	POSENC2EN	POSENC1EN	Res.	Res.	LPTIM1EN	Res.	Res.	I2C4EN	I2C3EN	LPUART1EN	Res.	TIM27EN	TIM26EN	TIM20EN	SBSEN	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0			0			0	0	0		0	0	0	0	
0x0AC	Reserved	Res.																															
0x0B0	RCC_AHB1LPENR	SRAM1LPEN	DCACHELPEN	ICACHELPEN	Res.	SRAM3LPEN	SRAM2LPEN	Res.	TZSC1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGLPEN	Res.	FMAC1LPEN	CORDIC1LPEN	Res.	CRCLPEN	Res.	Res.	Res.	FLASHLPEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2LPEN	GPDMA1LPEN
	Reset value	1	1	1		1	1		1							1		1	1		1				1							1	1

Table 132. RCC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B4	RCC_AHB2LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC3LPEN	ADC12LPEN	FMAC2LPEN	Res.	Res.	Res.	Res.	Res.	DAC10LPEN	DAC9LPEN	DAC8LPEN	DAC7LPEN	DAC6LPEN	DAC5LPEN	Res.	Res.	DAC2LPEN	DAC1LPEN
	Reset value															1	1	1						1	1	1	1	1	1			1	1
0x0B8	RCC_AHB3LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											1																					
0x0BC	RCC_AHB4LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XSPI1LPEN	Res.	Res.	Res.	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN	Res.	OTFDEC1LPEN	Res.	CCBLPEN	SAESLPEN	PKALPEN	RNGLPEN	HASHLPEN	AESLPEN
	Reset value												1				1	1	1	1	1	1	1	1		1		1	1	1	1	1	1
0x0C0	Reserved	Res.																															
0x0C4	RCC_APB1LLPENR	Res.	Res.	Res.	Res.	Res.	Res.	UART6LPEN	CRSLPEN	I3C1LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	WWDGLPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM11LPEN	TIM10LPEN	TIM9LPEN	TIM7LPEN	TIM6LPEN	Res.	Res.	TIM3LPEN	TIM2LPEN
	Reset value							1	1	1	1	1	1	1	1	1		1	1		1	1	1	1	1	1	1	1	1			1	1
0x0C8	RCC_APB1HLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	Res.	Res.	Res.	DTSLPEN	Res.	Res.	Res.	Res.
	Reset value									1														1						1			
0x0CC	RCC_APB2LPENR	Res.	TIM5LPEN	TIM4LPEN	Res.	Res.	Res.	Res.	USBLPEN	Res.	Res.	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	Res.	USART1LPEN	TIM8LPEN	SPI1LPEN	TIM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLAY1LPEN	Res.	Res.	Res.	HRTIM1LPEN
	Reset value		1	1					1						1	1	1		1	1	1	1								1				1
0x0D0	RCC_APB3LPENR	COMP8LPEN	COMP7LPEN	COMP6LPEN	COMP5LPEN	COMP4LPEN	COMP3LPEN	COMP2LPEN	COMP1LPEN	RTCAPBLPEN	VREFLPEN	Res.	Res.	Res.	OPAMP3LPEN	OPAMP2LPEN	OPAMP1LPEN	POSENC2LPEN	POSENC1LPEN	Res.	Res.	LPTIM1LPEN	Res.	Res.	I2C4LPEN	I2C3LPEN	LPUART1LPEN	Res.	TIM27LPEN	TIM26LPEN	TIM20LPEN	SBSLPEN	Res.
	Reset value	1	1	1	1	1	1	1	1	1	1				1	1	1	1	1			1			1	1	1		1	1	1	1	
0x0D4	Reserved	Res.																															
0x0D8	RCC_CCIPR1	CALCKEN	Res.	Res.	Res.	Res.	LPUART1SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	UART6SEL[2:0]			UART5SEL[2:0]			UART4SEL[2:0]			USART3SEL[2:0]			USART2SEL[2:0]			USART1SEL[2:0]		
	Reset value	0					0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 132. RCC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0DC	RCC_CCIPR2	Res.	DAC10SEL[2:0]			Res.	DAC9SEL[2:0]			Res.	DAC8SEL[2:0]			Res.	DAC7SEL[2:0]			Res.	DAC6SEL[2:0]			Res.	DAC5SEL[2:0]			DAC2SHSEL		DAC2SEL[2:0]		DAC1SHSEL	DAC1SEL[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0
0x0E0	RCC_CCIPR3	Res.	Res.	SYSTICKSEL[1:0]		Res.	LPTIM1SEL[2:0]			PLAY1DIV[3:0]				Res.	PLAY1SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI3SEL[2:0]			SPI2SEL[2:0]			SPI1SEL[2:0]		
	Reset value			0	0		0	0	0	0	0	0	0		0	0	0								0	0	0	0	0	0	0	0	0
0x0E4	RCC_CCIPR4	Res.	I2C4SEL[2:0]			Res.	I2C3SEL[2:0]			Res.	I2C2SEL[2:0]			Res.	I2C1SEL[2:0]			Res.	I3C1SEL[2:0]			Res.	Res.	USB5EL[1:0]		Res.	Res.	UCPD1SEL[1:0]		Res.	XSPI1SEL[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0			0	0			0	0		0	0	0
0x0E8	RCC_CCIPR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POSENC2SEL[2:0]			Res.	POSENC1SEL[2:0]			Res.	Res.	FDCANSEL[1:0]		Res.	Res.	Res.	Res.	ADCDIV[3:0]				Res.	ADCSEL[2:0]		
	Reset value										0	0	0		0	0	0			0	0					0	0	0	0		0	0	0
0x0EC	Reserved	Res.																															
0x0F0	RCC_RTCCR	Res.	Res.	Res.	Res.	LSIRDY	LSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCDRST	RTCEN	Res.	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
	Reset value					0	0										0	0							0	0		0	0	0	0	0	0	0
0x0F4	RCC_RSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	1	1			0																							
0x0F8 - 0x10C	Reserved	Res.																															
0x110	RCC_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVFSEC	HSI48SEC	Res.	Res.	PLL2SEC	PLL1SEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	Res.	HSESEC	HSISEC
	Reset value																				0	0			0	0	0	0	0	0		0	0
0x114	RCC_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
	Reset value																															0	0

Table 132. RCC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x118 - 0x1FC	Reserved	Res.																															
0x200	RCC_TESTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTREGEN
	Reset value																																0

Refer to *Section 2.3* for the register boundary addresses.

12 Clock recovery system (CRS)

This section applies only to STM32G55x/56x devices.

12.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from alternative synchronization or generated by the user software.

12.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity (see *Section 12.4.2: CRS internal signals*)
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

12.3 CRS implementation

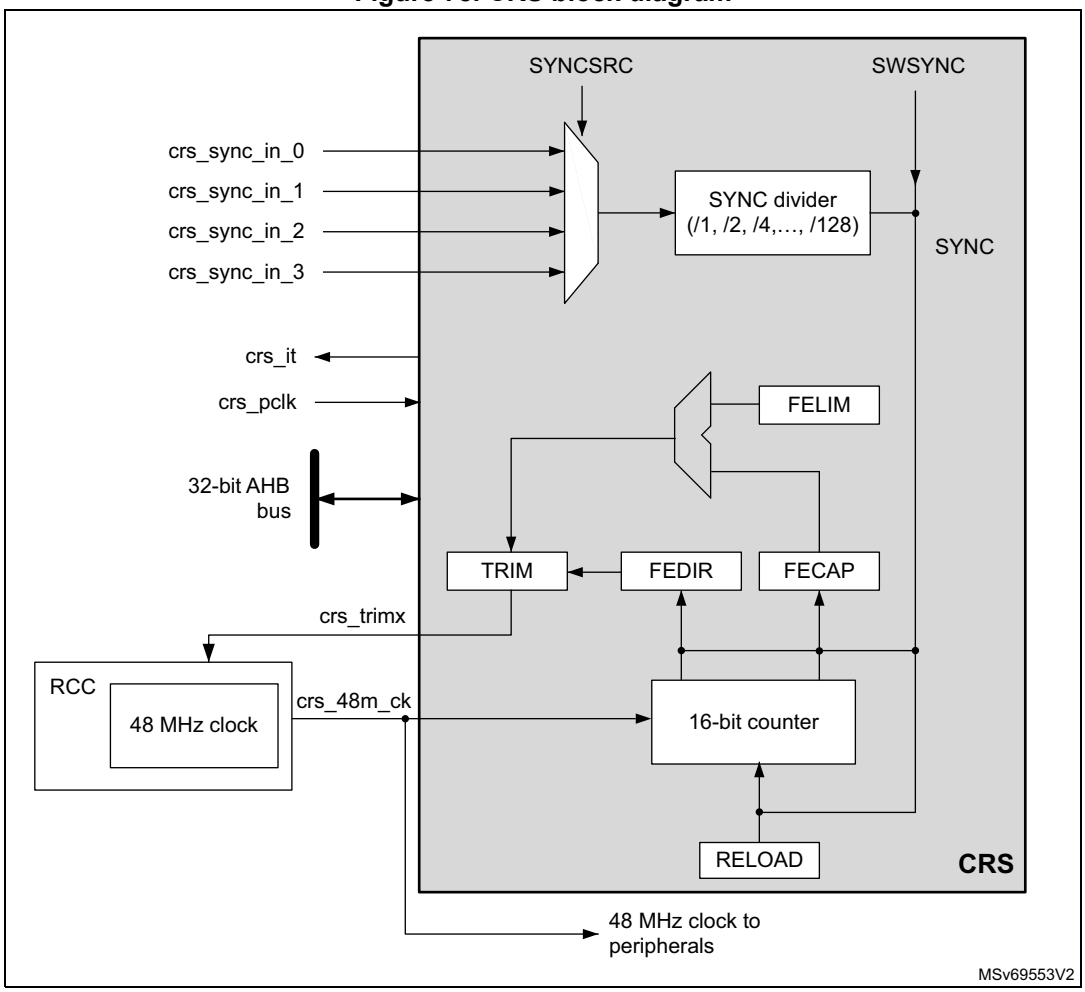
Table 133. CRS features

Feature	CRS
TRIM width	6 bits

12.4 CRS functional description

12.4.1 CRS block diagram

Figure 78. CRS block diagram



12.4.2 CRS internal signals

Below the list of CRS internal signals.

Table 134. CRS internal input/output signals

Internal signal name	Signal type	Description
crs_it	Digital output	CRS interrupt
crs_pclk	Digital input	AHB bus clock
crs_48m_ck	Digital input	HSI48 48 MHz clock
crs_trim[0:5]	Digital output	HSI48 oscillator smooth trimming value

Table 134. CRS internal input/output signals (continued)

Internal signal name	Signal type	Description
crs_sync_in_0	Digital input	SYNC signal source
crs_sync_in_1		
crs_sync_in_2		
crs_sync_in_3		

Table 135. CRS interconnection

Internal signal name	Description
crs_sync_in_0	GPIO selected as SYNC signal source
crs_sync_in_1	LSE selected as SYNC signal source
crs_sync_in_2	USB SOF selected as SYNC signal source (default)
crs_sync_in_3	Reserved

12.4.3 Synchronization input

The CRS synchronization source (crs_sync_in_x) can be selected through SYNCSRC[1:0] bitfield of CRS_CFGR register (refer to *Section 12.4.2: CRS internal signals* for the possible sources). For a better robustness of the crs_sync_in_x input, a simple digital filter (2 out of 3 majority votes, sampled by the 48 MHz clock) is implemented to filter out glitches. In addition, the source signal has a configurable polarity (selected through SYNCPOL bit of CRS_CFGR). The signal can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to CRS_CFGR register.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

For more information on the CRS synchronization source configuration, refer to *CRS configuration register (CRS_CFGR)*.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

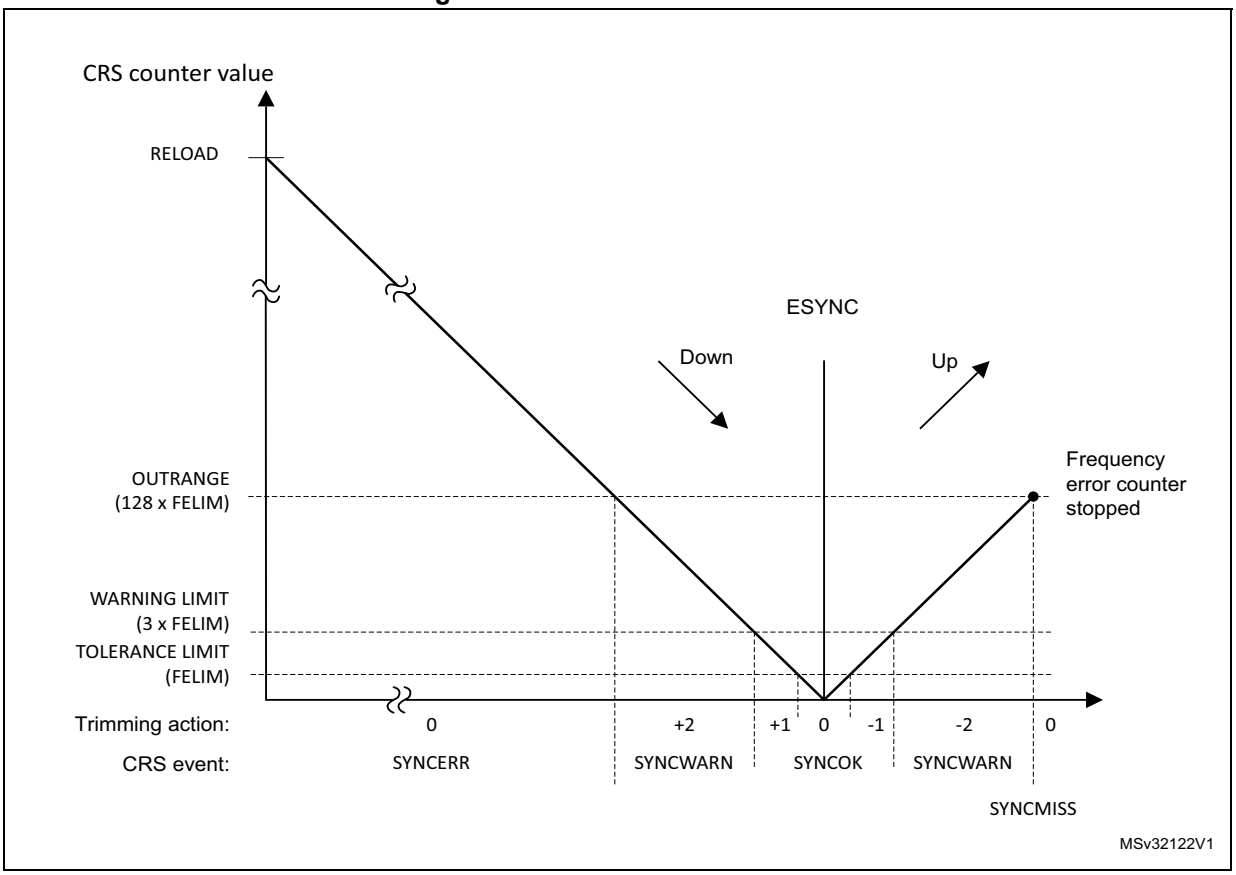
12.4.4 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual

frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 79. CRS counter behavior



12.4.5 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 × FELIM value
- OUTRANGE (error limit), defined as 128 × FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated

  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

12.4.6 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

$$RELOAD = (f_{TARGET} / f_{SYNC}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to $(f_{TARGET} / f_{SYNC}) = 48000$, and to a typical trimming step size of 0.14%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

12.5 CRS in low-power modes

Table 136. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.
Shutdown	The peripheral is powered down and must be reinitialized after exiting Shutdown mode.

12.6 CRS interrupts

Table 137. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

12.7 CRS registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

12.7.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]						SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNC IE	ERR IE	SYNC WARNIE	SYNC OKIE
		rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **TRIM[5:0]**: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the oscillator frequency.

The default value of the HSI48 oscillator smooth trimming is 32, for a complete TRIM range of 64 steps.

The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 12.4.5* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled

1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable

0: SYNC warning (SYNCWARNF) interrupt disabled

1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable

0: SYNC event OK (SYNCOKF) interrupt disabled

1: SYNC event OK (SYNCOKF) interrupt enabled

12.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]							
rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.

0: SYNC active on rising edge (default)

1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source (see *Section 12.4.2: CRS internal signals*).

00: crs_sync_in_0 selected as SYNC signal source

01: crs_sync_in_1 selected as SYNC signal source

10: crs_sync_in_2 selected as SYNC signal source

11: crs_sync_in_3 selected as SYNC signal source

Note: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the 48 MHz clock on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.

000: SYNC not divided (default)

001: SYNC divided by 2

010: SYNC divided by 4

011: SYNC divided by 8

100: SYNC divided by 16

101: SYNC divided by 32

110: SYNC divided by 64

111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to *Section 12.4.5* for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to *Section 12.4.4* for more details about counter behavior.

12.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to *Section 12.4.5* for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

0: Up-counting direction, the actual frequency is above the target

1: Down-counting direction, the actual frequency is below the target

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No trimming error signaled

1: Trimming error signaled

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC missed error signaled

1: SYNC missed error signaled

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC error signaled

1: SYNC error signaled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

0: No expected SYNC signaled

1: Expected SYNC signaled

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signaled

1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

0: No SYNC warning signaled

1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

0: No SYNC event OK signaled

1: SYNC event OK signaled

12.7.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNC WARNC	SYNC OKC
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCC**: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag
Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag
Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

12.7.5 CRS register map

Table 138. CRS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value	0x0000 2000																															
0x04	CRS_CFGR	SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	CRS_ISR	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0					0	0	0	0
0x0C	CRS_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

13 General-purpose I/Os (GPIO)

13.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16-bit reset register (GPIOx_BRR), and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL), a secure configuration register (GPIOx_SECCFGR).

13.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bit-wise write access to GPIOx_ODR
- Lock mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions
- TrustZone® security support
- I/Os state retention during Standby mode

13.3 GPIO functional description

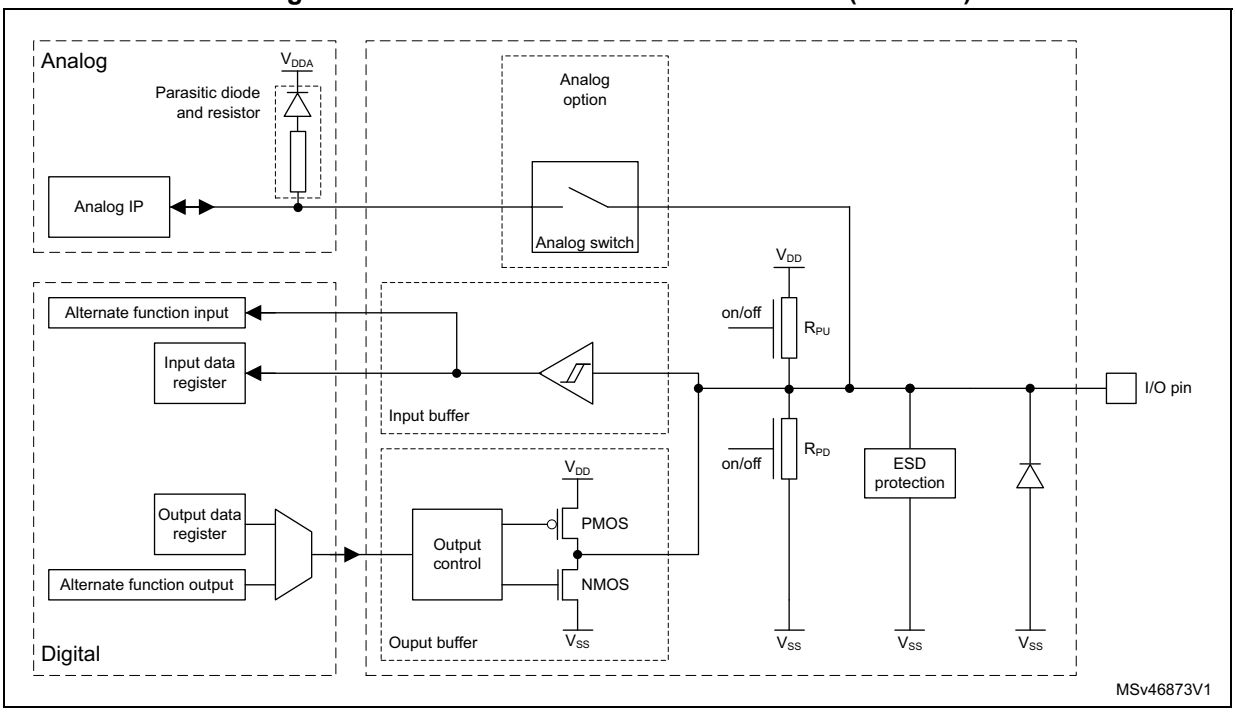
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words, or bytes. The GPIOx_BSRR and GPIOx_BRR registers allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 80 shows the basic structure of a 3- or 5-V tolerant GPIO (TT or FT). *Table 139* gives the possible port bit configurations.

Figure 80. Structure of 3- or 5-V tolerant GPIO (TT or FT)



Note: On a TT GPIO, the analog switch is not present, it is replaced by a direct connection.

Caution: When the analog switch is enabled, it connects analog blocks, which does not allow five-volt tolerance.

Table 139. Port bit configuration(1)

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]	PUPD(i)[1:0]		I/O configuration	
01	0	SPEED[1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Table 139. Port bit configuration[1] (continued)

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]		PUPD(i)[1:0]		I/O configuration	
10	0	SPEED[1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Analog I/O	Floating
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Analog I/O	PD
	x	x	x	1	1	Reserved	

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

13.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO in floating state no pull-up/pull-down

BOOT0 is in input mode during the reset until at least the end of the option byte loading phase (see *Section 13.3.17*).

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, the high level is high-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not, depending upon the value in the GPIOx_PUPDR register.

13.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset, the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user must proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**: configure the desired I/O as output, input, or analog in the GPIOx_MODER register.
- **Alternate function**:
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down, and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDR registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
  - The Cortex-M33 output EVENTOUT signal can be output as alternate function on I/O. An event can be signaled through the configured pin after executing SEV instruction.
- **Additional functions:**
  - For the ADC and DAC, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC and DAC registers.
  - For the additional functions like RTC, WKUPx, and oscillators, configure the required function in the related RTC, PWR, and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

13.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

13.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (*GPIO port input data register (GPIOx_IDR) (x = A to H)* and *GPIO port output data register (GPIOx_ODR) (x = A to H)*).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data entered through the I/Os are stored into the input data register (GPIOx_IDR), a read-only register.

13.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

13.3.6 GPIO locking mechanism

The GPIO control registers can be frozen by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to the bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence can be performed only using a word (32-bit long) access to the GPIOx_LCKR register, because GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

13.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that some peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus

select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

13.3.8 External interrupt/wake-up lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output, or alternate function mode (the port must not be configured in analog mode). Refer to *Section 19: Extended interrupts and event controller (EXTI)*.

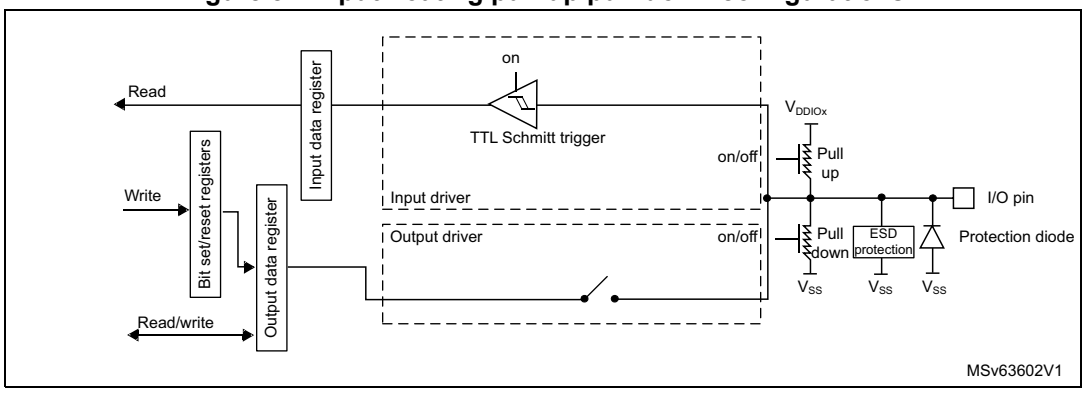
13.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

Figure 81 shows the input configuration of the I/O port bit.

Figure 81. Input floating/pull-up/pull-down configurations



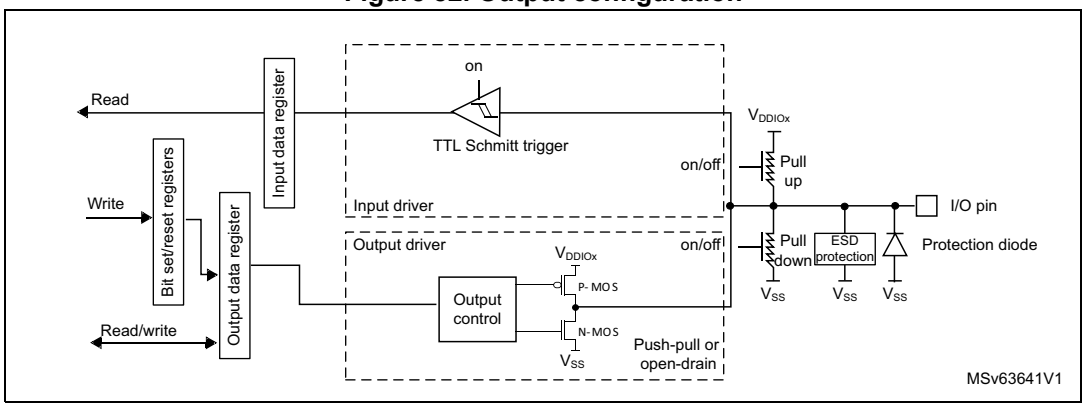
13.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register leaves the port in high-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

Figure 82 shows the output configuration of the I/O port bit.

Figure 82. Output configuration



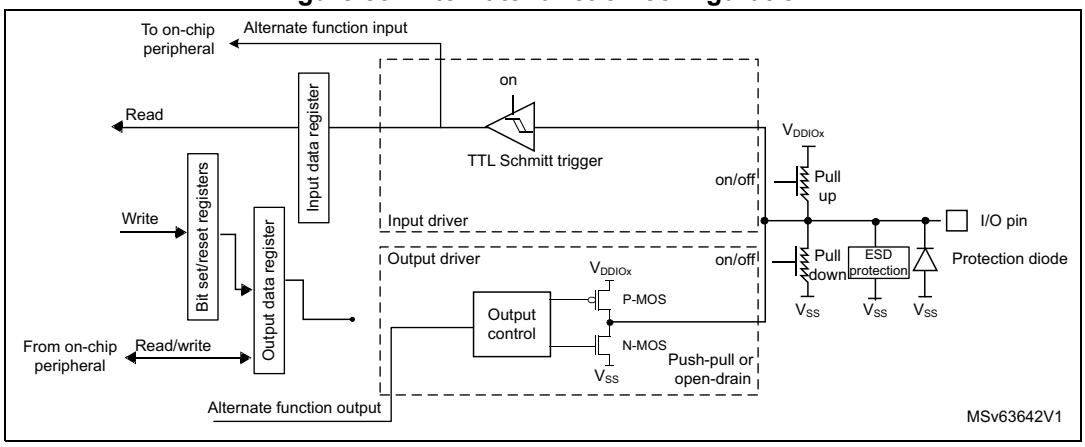
13.3.11 Alternate function configuration

When the I/O port is programmed as an alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.

Figure 83 shows the alternate function configuration of the I/O port bit.

Figure 83. Alternate function configuration



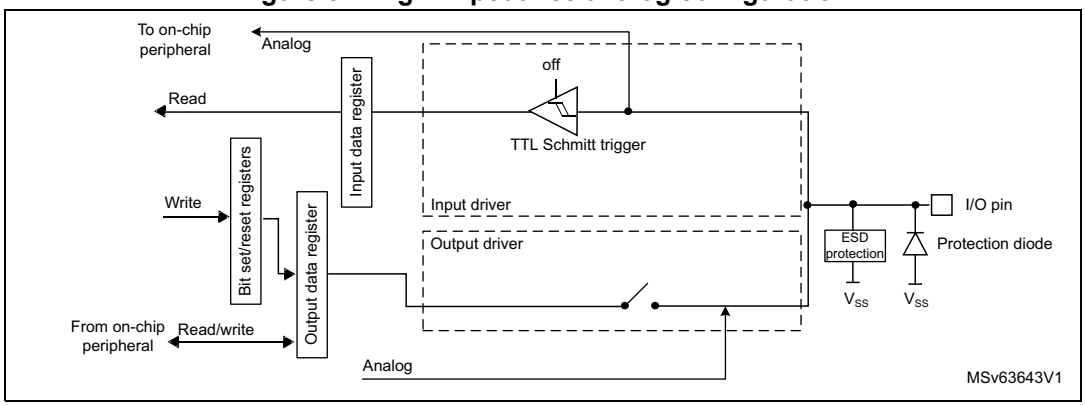
13.3.12 Analog configuration

When the I/O port is programmed as an analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up is disabled by hardware. The weak pull-down is configurable.
- Read access to the input data register gets the value 0.

Figure 84 shows the high-Z, analog-input configuration of the I/O port bits.

Figure 84. High-impedance analog configuration



13.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched on (by setting the HSEON or LSEON bit in the RCC_CSR register), the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input, and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

OSC32_IN and OSC32_OUT pins have limited driving capability and speed (see device datasheet for the data).

13.3.14 Using the GPIO pins in the RTC domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 49.3: RTC functional description*.

13.3.15 Privileged and unprivileged modes

All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or nonsecure).

13.3.16 SMBUS 3.3 compliant mode for I2C pins

Some I/Os have the capability to be compliant with SMBUS 3.3 version. See device datasheet pinout table for a list of GPIOs that support SMBUS 3.3 compliant mode. To enable SMBUS 3.3 compliant mode on those pins, the pin must be configured in AF mode for the I2Cx peripheral and the corresponding I2CxSMB3EN bit must be set (in SBS_PMCR register).

13.3.17 I/Os state retention during standby mode

In the Standby mode, the I/Os are by default in floating state.

If the IORETEN or JTAGIORETEN bit in the PWR_IORETR register is set, the I/Os state is sampled during standby entry. The state of I/Os is applied to the pin via pull-up and pull-down resistors. The pull-up and pull-down resistors remains applied after Standby wake-up until the IORETEN/JTAGIORETEN bit in the PWR_IORETR register is cleared by the software.

13.3.18 TrustZone security

The TrustZone security is activated by the TZEN option byte in the FLASH option byte register. When the TrustZone is active (TZEN = 0xB4), each I/O pin of the GPIO port can be individually configured as secure through the GPIOx_SECCFGR register.

When the selected I/O pin is configured as secure, its corresponding configuration bits for alternate function, mode selection, I/O data are secure against a nonsecure access. In case of nonsecure access, these bits are RAZ/WI.

The I/Os with peripheral functions are also conditioned by the peripheral security configuration (see *Section 5: Global TrustZone® controller (GTZC)* for more details):

- For peripherals for which the I/O pin selection is done through alternate functions registers: if the peripheral is configured as secure, it cannot be connected to a nonsecure I/O pin. If this is not respected, the input data to the secure peripheral is

forced to 0 (I/O input pin value is ignored) and the output pin value is forced to 0, thus avoiding any secure information leak through nonsecure I/Os.

- For I/Os with analog switches, directly controlled by peripherals (such as ADC for instance): If the I/O is secure, the I/O analog switch cannot be controlled by a nonsecure peripheral. If this is not respected, the switch remains open. This prevents the redirection of secure data to a nonsecure peripheral or I/O through the analog path. Refer to *Section 3: System security* for more details.
- Some of the paths between I/Os “additional functions” and peripherals are not blocked if the I/O is secure and the peripheral is nonsecure. Therefore, it is recommended to configure those peripherals as secure even when not used by the application. Refer to *Section 3: System security* for the list of concerned peripherals. When the path has a security control, it follows the same rule as I/O selection through alternate functions.

Refer to the device pins definition table in the datasheet for more information about peripherals alternate functions and additional functions mapping.

After reset, all GPIO ports are secure.

The table below gives a summary of the I/O port-secured bits following the security configuration bit in the GPIO_SECCFGR register. When the I/O bit port is configured as secure:

- Secured bits: read and write operations are only allowed by a secure access. Non secure-read or write accesses on secured bits are RAZ/WI. There is no illegal access event generated.
- Nonsecure bits: no restriction. Read and write operations are allowed by both secure and nonsecure accesses.

When the TrustZone security is disabled (TZEN = 0xC3 in FLASH_OPTSR2 register), all registers bits are nonsecure. The GPIOx_SECCFGR register is WI and its content has no effect.

Table 140. GPIO secured bits

Secure configuration bit	Secured bit	Register name	Nonsecure access on secure bits
SECy = 1 in GPIOx_SECCFGR(1)	MODEy[1:0]	GPIOx_MODER	RAZ/WI
	OTy	GPIOx_OTYPER	
	OSPEEDy[1:0]	GPIOx_OSPEEDR	
	PUPDy[1:0]	GPIOx_PUPDR	
	IDy	GPIOx_IDR	
	ODy	GPIOx_ODR	
	BSy and BRy	GPIOx_BSRR	
	LCKy	GPIOx_LCKR	
	BRy	GPIOx_BRR	
	AFSELy[3:0]	GPIOx_AFRH	
		GPIOx_AFRL	

1. x = A to H

13.3.19 I/O compensation cell

The I/O commutation slew rate (tfall / trise) can be adapted by software depending on process, voltage and temperature conditions, to reduce the I/O noise on the power supply. Refer to *Section 14: System configuration, boot, and security (SBS)* for more details.

13.4 GPIO registers

This section gives a detailed description of the GPIO registers.

The peripheral registers can be written in word, half-word, or byte mode.

13.4.1 GPIO port mode register (GPIOx_MODER) (x = A to H)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFFFC FEBF (for port B)

Reset value: 0xFFFF FFFF (for port C) (0xFFC3 FFFF for STM32G53x/54x devices)

Reset value: 0xFFFF FFFF (for port D) (0x03FF 0000 for STM32G53x/54x devices)

Reset value: 0xFFFF FFF0 (for port E) (0xFC00 0030 for STM32G53x/54x devices)

Reset value: 0xFFFF C3F3 (for port F) (0x003C 0330 for STM32G53x/54x devices)

Reset value: 0x3FFC 0FF0 (for port G) (0x0000 00F0 for STM32G53x/54x devices)

Reset value: 0x0000 0C0F (for port H) (0x0000 0000F for STM32G53x/54x devices)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to H)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to H)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very-high speed

Note: Refer to the device datasheet for the frequency specifications, the power supply, and the load conditions for each speed.

The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to H)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.5 GPIO port input data register (GPIOx_IDR) (x = A to H)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.6 GPIO port output data register (GPIOx_ODR) (x = A to H)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A to H).

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to H)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 15:0 **BSy:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to H)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

- LOCK key write sequence:

WR LCKR[16] = 1 + LCKR[15:0]

WR LCKR[16] = 0 + LCKR[15:0]

WR LCKR[16] = 1 + LCKR[15:0]

- LOCK key read

RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the LOCK.

After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0

0: Port configuration not locked

1: Port configuration locked

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to H)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to H)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to H)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRy:** Port x reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.12 GPIO secure configuration register (GPIOx_SECCFGR) (x = A to H)

When the system is secure (TZEN = 0xB4), this register provides write access security and can be written only by a secure access. It is used to configure a selected I/O as secure. A nonsecure write access to this register is discarded.

When the system is not secure (TZEN = 0xC3), this register is WI and its content has no effect.

Address offset: 0x30

Reset value: 0x0000 FFFF (for port A)

Reset value: 0x0000 FEFF (for port B)

Reset value: 0x0000 FFFF (for port C) (0x0000 F9FF for STM32G53x/54x devices)

Reset value: 0x0000 FFFF (for port D) (0x0000 1F00 for STM32G53x/54x devices)

Reset value: 0x0000 FFFC (for port E) (0x0000 E004 for STM32G53x/54x devices)

Reset value: 0x0000 FF9D (for port F) (0x0000 0614 for STM32G53x/54xdevices)

Reset value: 0x0000 7E3C (for port G) (0x0000 000C for STM32G53x/54x devices)

Reset value: 0x0000 0023 (for port H) (0x0000 00003 for STM32G53x/54x devices)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECy**: I/O pin of Port x secure bit enable y (y = 15 to 0)

These bits are written by software to enable or disable the I/O port pin security.

0: The I/O pin is nonsecure

1: The I/O pin is secure. Refer to *Table 140* for all corresponding secured bits.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.13 GPIO register map

Table 141. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOx_MODER (x = A to H)**	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value for port A	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
	Reset value for ports C...H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**GPIOx_OTYPER (x = A to H)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_ OSPEEDR (x = A to H)**	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value for port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	Reset value for ports C...H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOx_PUPDR (x = A to H)**	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value for port A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Reset value for ports C...H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR (x = A to H)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR (x = A to H)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR (x = A to H**	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR (x = A to H)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL (x = A to H)**	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH (x = A to H)**	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 141. GPIO register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**GPIOx_BRR (x = A to H)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**GPIOx_ SECCFGR (x = A to H**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3* for the register boundary addresses.

14 System configuration, boot, and security (SBS)

14.1 SBS introduction

The STM32G56x/55x/54x/53x devices feature a set of configuration registers located in the SBS. On top of various device configurations, this peripheral controls key boot and security features, including debug control and secure storage control.

14.2 SBS main features

- System configuration
  - Manage safety feature
  - Enable/disable the FMP (Fast-mode Plus) high-drive capability of some I/Os
  - Manage the I/O compensation cell
  - Configure register security access
- Boot control
  - Upon system reset, configure the Cortex-M33 boot address and the temporal isolation level depending on the current configuration such as PRODUCT_STATE (sbs_product_state), BOOT_UBE (sbs_irot_select), or TZEN (sbs_tzen).
  - Manage the temporal isolation feature implemented thanks to the hide protect level (HDPL) monotonic counter
- Debug control
  - Control the opening of the device debug interface, ensuring the sequencing of events that guarantee the device security
- Hardware secure storage control
  - Control the secure storage selections: OBK-HDPL selection (selects the area of secure storage of the flash memory and selects the corresponding DHUK in SAES), EPOCH selection

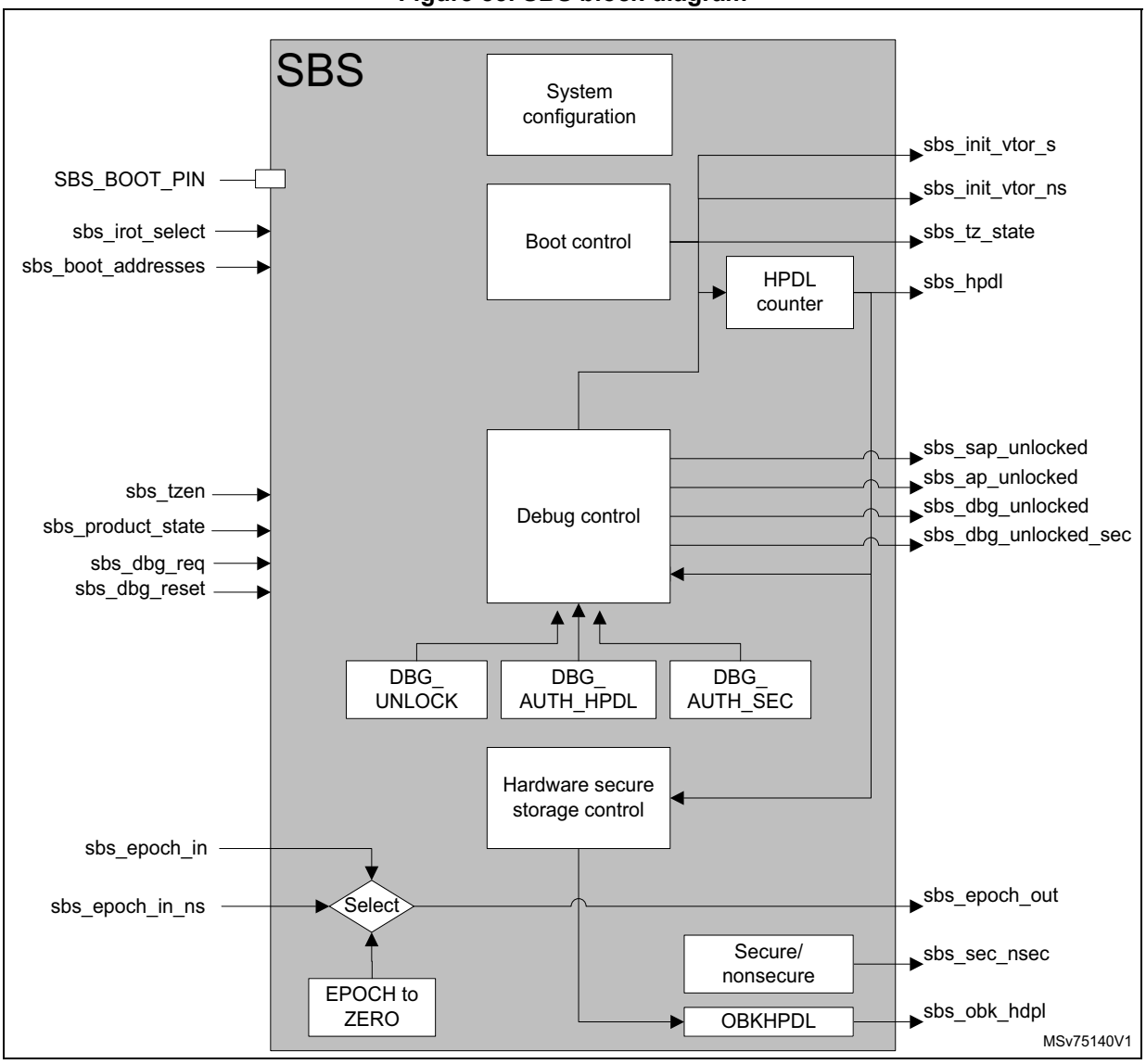
14.3 SBS functional description

14.3.1 SBS block diagram

Figure 85 shows the SBS block diagram, including the four main functions:

- System configuration
- Boot control
- Debug control
- Hardware secure storage control

Figure 85. SBS block diagram



14.3.2 SBS signals

Table 142 details the user relevant internal signals that interface the SBS.

Table 142. SBS internal input/output signals

Signal name	Type	Description
BOOT0	Input	Select booting on user flash memory or bootloader when TrustZone is disabled (TZEN = 0xC3), or on RSS when TrustZone is enabled (TZEN = 0xB4).
sbs_irot_select		Signal based on BOOT_UBE option byte to select the iROT between STiRoT and user flash memory (OEMiRoT).
sbs_tzen		Signal based on TZEN option byte to activate/deactivate the TrustZone
sbs_boot_addresses		List of addresses defined by the flash memory: – BOOTADD: nonsecure boot address – SBOOTADD: secure boot address
sbs_product_state		Signal based on PRODUCT_STATE option byte to activate the different security mechanisms depending on the product use. Expected values are described in *Section 7: Embedded flash memory (FLASH)*.
sbs_dbg_req		Launch the debug authentication protocol when booting.
sbs_init_vtor_s	Output	Vector address for Cortex-M33 secure entry point.
sbs_init_vtor_ns		Vector address for Cortex-M33 nonsecure entry point.
sbs_tz_state		Inform the Cortex-M33 on the secure state of the core.
sbs_hdpl		HDPL (temporal isolation level, ID of the boot level, used to isolate boot levels) This signal reflects a monotonic counter that can be incremented from 0 to 3, and that is reset to 0 only on software reset or POR.
sbs_ap_unlocked		Control the Cortex-M33 access port.
sbs_sap_unlocked		Control the AHB access port.
sbs_dbg_unlocked		Unlock the debug (when set to 1) for the Cortex-M33 nonsecure part.
sbs_dbg_unlocked_sec		Unlock the debug (when set to 1) for the Cortex-M33 secure part.
sbs_dbg_reset	Input	Control the reset of the debug authentication configuration to be done with a system reset or a power-on reset. The configuration is done through the DBGMCU using the DCRT bitfield of DBGMCU_CR register.
sbs_obk_hdpl	Output	Select secure storage domain (OBK-HDPL) for current HDPL, or greater ones to allow provisioning.
sbs_epoch_out		EPOCH counters (NS_EPOCH and SEC_EPOCH, inputs from flash memory) are used to manage the REPLAY protection.
sbs_sec_nsec		Reflect secure/nonsecure selection for the secure storage.

14.3.3 SBS reset and clocks

The SBS configuration port is clocked by the AHB bus clock. There is a general reset and a debug configuration reset controlled in DBGMCU.

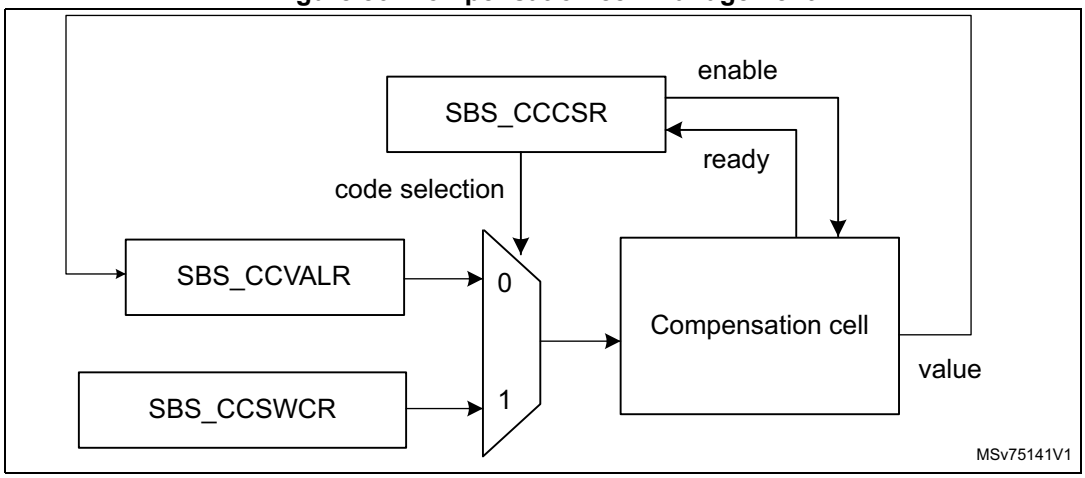
14.3.4 SBS system configuration

SBS I/O compensation cell management

The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS) that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the output impedance in the I/O buffer, and the slew rate of the I/O commutation (the t_{fall} and t_{rise} time) in order to reduce the I/O noise on power supply.

As shown in *Figure 86*, the compensation cell is split in two blocks: one block to provide an optimal code for the current PVT, and one block to drive the block controlled by the software.

Figure 86. Compensation cell management

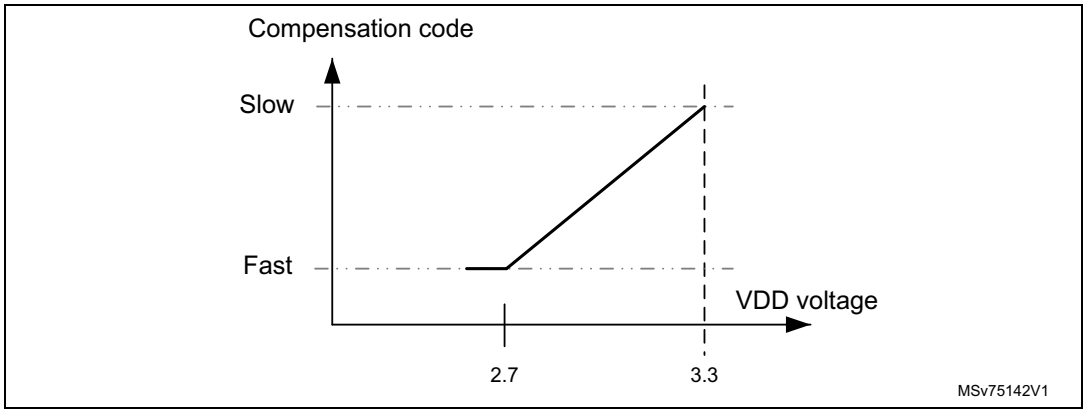


The compensation cell value can be read when the READY flag is set in SBS_CCCSR. With CS1 in SBS_CCCSR, the application can select the value to apply between two options: the code from the cell or the code from SBS_CCSWCR.

One compensation cell is embedded in STM32G56x/55x/54x/53x and dedicated to the I/Os supplied by the V_{DD} power rail.

By default, the compensation cells are disabled, and a fixed code is applied to all the I/Os.

Figure 87. Compensation cell use



SBS TrustZone security and privilege

- SBS TrustZone security

  When the TrustZone security is activated, the SBS is able to protect secure registers from being modified by nonsecure accesses.

  The TrustZone security is activated by the TZEN option byte in FLASH_OPTSR2_PRG.

  A nonsecure read/write access to a secured register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the SBS illegal access event is enabled in the GTZC.

- Privileged/unprivileged mode

  The SBS registers can be read and written by privileged and unprivileged accesses except the SBS registers for CPU configuration:

  – SBS_CSLCKR, SBS_FPUIMR, SBS_CNSLCKR, SBS_EPOCHSELCR, SBS_ECCNMIR, SBS_ECCNMIR, SBS_SWAPLOCKR and SBS_FBMODER

  – FPUSEC in SBS_SECCFGR

  An unprivileged access to a privileged register is RAZ/WI.

14.3.5 SBS SMBUS 3.3 control

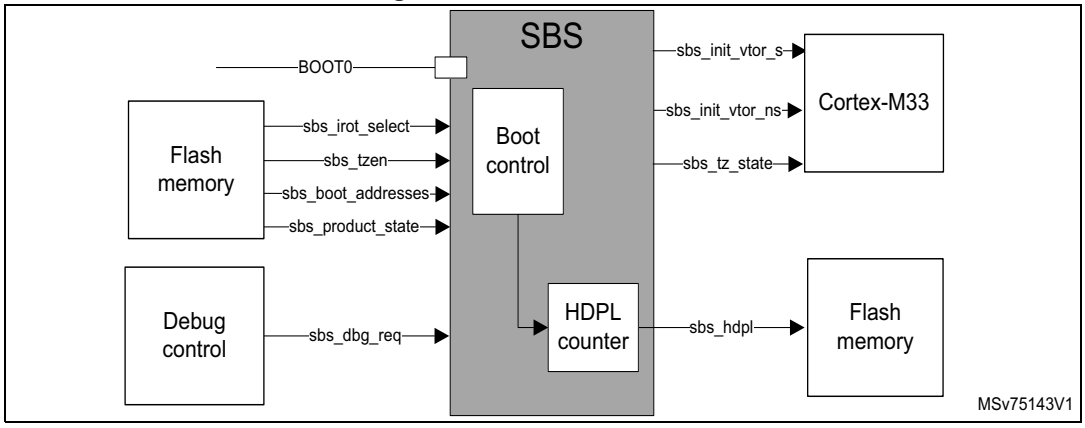
The SBS can be used to control the I/Os which are compliant with SMBUS revision 3.3 specification. SMBUS 3.3 compliant mode on specific pins (see device datasheet pinout table) is controlled by corresponding I2CxSMB3EN bit (in SBS_PMCR register).

14.3.6 SBS boot control

The SBS can be used to control the boot entry points considering the product settings. The main boot control actions are listed below:

- Run product with or without TrustZone enabled.
- Select between STiRoT or OEMiRoT.
- Boot when launching a debug authentication sequence.
- Select boot between the bootloader or the user flash memory boot.
- Initialize the HDPL boot value.

Figure 88. SBS boot control



The boot configurations are selected considering the product settings:

- BOOT0: to select booting on user flash memory or RSS (root secure services)
- BOOT_UBE option byte to select the iRoT between STiRoT and OEMiRoT
- TZEN option byte to activate/deactivate the TrustZone
- sbs_boot_addresses: list of addresses defined by the flash memory:
  - BOOTADD: nonsecure boot address
  - SBOOTADD: secure boot address
- PRODUCT_STATE: option byte to activate the different security mechanisms depending on the product use. Expected values are described in *Section 7: Embedded flash memory (FLASH)*.
- sbs_dbg_req: used to launch the debug authentication protocol when booting.

The boot control logic sets the following data:

- sbs_init_vtor_s: vector address for Cortex-M33 secure entry point
- sbs_init_vtor_ns: vector address for Cortex-M33 nonsecure entry point
- sbs_tz_state (secure/nonsecure): informs the Cortex-M33 on the secure state of the core.
- HDPL

SBS HDPL (temporal isolation level) management

The HDPL is a monotonic counter incremented during the boot stages. The HDPL is reset to its default value only after a power-on or a system reset. This default value (0 or 1) depends on the device life cycle, as defined in boot logic.

The devices use HDPL information to automatically isolate code and its associated secrets (like keys) during the boot process. Incrementing HPDL ensures that private code and data for one boot stage cannot be directly accessible from later boot stages.

The HDPL is used by the user flash memory. See *Section 7: Embedded flash memory (FLASH)* for more details. The HDPL can take values from 0 to 3. When reaching 3, HDPL keeps this value until reset. The current HDPL value is readable in HDPL bitfield in SBS_HDPLSR.

To increment the HDPL by one, the application must write 0x6A to INCR_HDPL in SBS_HDPLCR. After such increment, and before doing any subsequent action, the user must check that the HDPL has effectively been incremented by reading SBS_HDPLSR.

Table 143. HDPL encoded values

HDPL	Code
0	0xB4
1	0x51
2	0x8A
3	0x6F
	All other values

Table 144. SBS boot logic

Inputs						Outputs			
sbs_ product_ state	**sbs_dbg_req**	**sbs_tzen**	**sbs_irotselect**	**BOOT0**	**sbs_boot _addresses**	**sbs_init_vtor_s**	**sbs_init_vtor_ns**	**sbs_hdpl**	**sbs_tz_state**
Any except Locked	1	x	x	x	BOOT_ DBG_AUTH_ADD	BOOT_DBG_ AUTH _ADD	x	1	Secure
Open	0	0	x	0	BOOTADD	x	BOOTADD	1	Nonsecure
	0	0	x	1	BOOT_ST_RSS _ADD	BOOT_ST_R SS_ADD	x	0	Secure
	0	1	x	0	SBOOTADD	SBOOT ADD	x	1	secure
	0	1	x	1	BOOT_ST_RSS _ADD	BOOT_ST_R SS_ADD	x	0	Secure
Provisioning	0	x	x	x	BOOT_ST_RSS _ADD	BOOT_ST_R SS_ADD	x	0	Secure
iRoT-Provisioned	0	0	x	x	BOOTADD	x	BOOTADD	1	Nonsecure
	0	1	0	x	BOOT_ST_iROT _ADD	BOOT_ST_ iROT_ADD	x	1	Secure
	0	1	1	x	SBOOTADD	x	SBOOTADD	1	Secure
TZ-closed	-	1	0	x	BOOT_ST_iROT _ADD	BOOT_ST_ iROT_ADD	x	1	Secure
	-	1	1	x	SBOOTADD	x	SBOOTADD	1	Secure
Closed	0	0	x	x	BOOTADD	x	BOOTADD	1	Nonsecure
	0	1	0	x	BOOT_ST_ iROT_ADD	BOOT_ST_ iROT_ADD	x	1	Secure
	0	1	1	x	SBOOTADD	x	SBOOTADD	1	Secure
Locked	0	0	x	x	BOOTADD	x	BOOTADD	1	Nonsecure
	0	1	0	x	BOOT_ST_ iROT_ADD	BOOT_ST_ iROT_ADD	x	1	Secure
	0	1	1	x	SBOOTADD	x	SBOOTADD	1	Secure

Table 144. SBS boot logic (continued)

Inputs						Outputs			
sbs_ product_ state	**sbs_dbg_req**	**sbs_tzen**	**sbs_irotselect**	**BOOT0**	**sbs_boot _addresses**	**sbs_init_vtor_s**	**sbs_init_vtor_ns**	**sbs_hdpl**	**sbs_tz_state**
Regression	0	0	x	x	BOOT_ DBG_AUTH_ADD	BOOT_ DBG_AUTH_ ADD	x	0	Secure
	0	1	x	x	BOOT_ DBG_AUTH_ADD	BOOT_ DBG_AUTH_ ADD	x	0	Secure
NS-Regression	0	1	x	x	BOOT_ DBG_AUTH_ADD	BOOT_ DBG_AUTH_ ADD	x	0	Secure

14.3.7 SBS debug control

The SBS debug control is used to manage debug opening, taking care on the product context (PRODUCT_STATE, TZEN, HDPL) on register settings, or through a debug authentication control.

When the debug is forbidden, the mailbox access port, Cortex-M33 access port, and CPU debug interface are locked. In this situation, the debugger cannot access the CPU and no effective debug can be done. Refer to the*Section 60: Debug support (DBG)* for more details.

Authenticated debug sequence

1. The external host requests to launch the debug authentication protocol via the DBGMCU access port mailbox. The rest of the device is kept under reset.
2. SBS selects the STMicroelectronics RSS-DA (debug authentication library) boot address, and requests the CPU to be released from reset.
3. The CPU running RSS-DA library executes the debug authentication protocol in the system flash memory. If the device is closed, the access port mailbox is closed until RSS-DA acknowledges the authentication sequence start request.
4. There are two authentication methods:
   – Certificate: As soon as the device fully verifies a debug certificate chain:

     If the certificate concerns a debug permission, then the STRSS-DA programs the debug opening of the Arm® Cortex-M33. Note that in case the TrustZone is

disabled (TZEN = 0xC3), the nonsecure only debug opening and secure/nonsecure debug opening have the same effect.

Alternatively, the certificate can authorize partial or full regression that allows debugging on a regressed part.

– Password: This method only allows the full regression of the product to be controlled.

5. Above reopenings are effective only when HDPL in SBS_HDPLSR has a value equal or superior to the value programmed in DBG_AUTH_HDPL in SBS_DBGCR. In case of authentication failure, the user is informed through the host interface.

*Note: The debug authentication library in system flash memory is available only when HDPL = 0 or 1 in SBS_*HDPL*SR. Only this library can perform the steps 3 and 4 described above.*

Debug reset

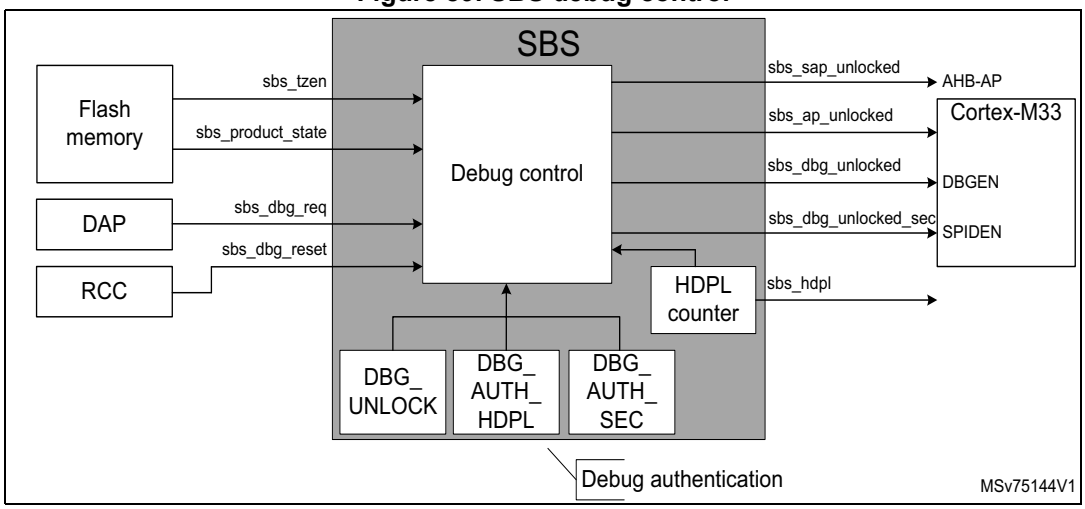
The debug opening can have the configuration to be reset by system or power-on reset, depending on a DBGMCU register field.

Debug locking

The debug configuration can be locked thanks to DBGCFG_LOCK in SBS_DBGLOCKR. SBS_DBGCR and SBS_DBGCR2 is then no longer writable.

When DBGCFG_LOCK is set to 1, it can be reset only by system or power-on reset. The configuration is done through the DBGMCU using the DCRT field of DBGMCU_CR.

Figure 89. SBS debug control



Inputs used to control the debug opening:

- sbs_dbg_req: input signal that a host requests to launch the debug authentication protocol. When this signal is set, the boot address is set to launch the debug authentication library.
- sbs_tzen: inform on the selected platform configuration related to TrustZone activation. When TZEN value is set to 0 (0xC3), the platform does not support TrustZone and the

simplified life cycle is proposed. See *Section 7: Embedded flash memory (FLASH)* for more details.

- sbs_product_state: provide the current PRODUCT_STATE (from flash memory). See *Section 7: Embedded flash memory (FLASH)* for more details.
- sbs_dbg_reset: reset information coming from the RCC, and reset SBS_DBGCR and SBS_CBGCR2 if these registers are configured to be reset.

Configuration

- DBG_UNLOCK in SBS_DBGCR: debug unlock when DBG_AUTH_HDPL is reached.
- AP_UNLOCK in SBS_DBGCR: access port unlock
- SEC_AP_UNLOCK in SBS_DBGCR: Arm® Cortex®-M33 (AHB_AP) secure access port unlock
- DBG_AUTH_HDPL in SBS_DBGCR: authenticated debug temporal isolation level. Define the HDPL value from which the debug can be opened. Value can only be from 1 to 3.
- DBG_AUTH_SEC in SBS_DBGCR: specify if the debug reopening is for nonsecure only or for both (secure and nonsecure).
- DBGC_LOCK in SBS_DBGLOCKR: lock the current debug configuration (released only by a reset).

Outputs

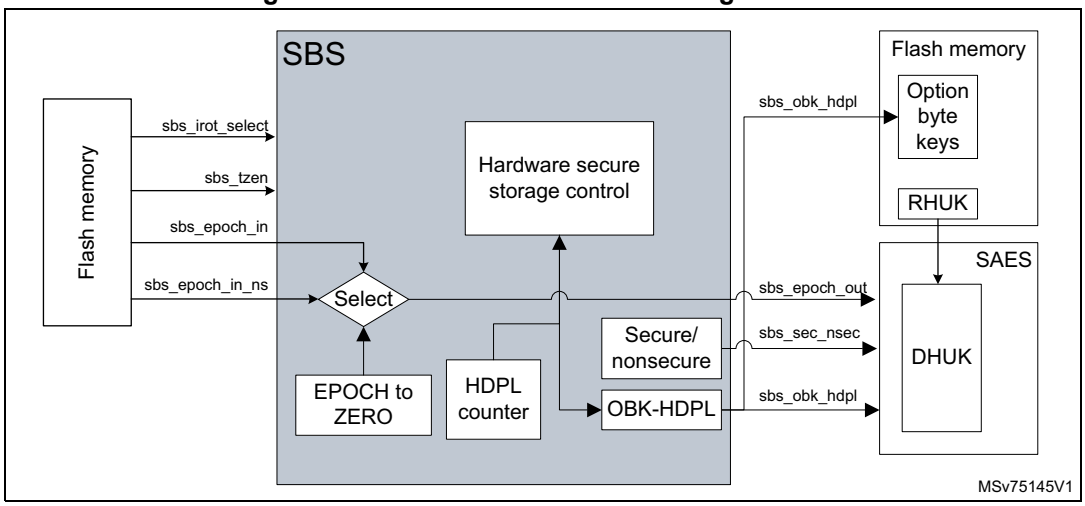
- sbs_sap_unlocked: signal controlling the AHB access port
  - 1: 0xB4
  - 0: all other codes
- sbs_ap_unlocked: signal controlling the Cortex-M33 access port
  - 1: 0xB4
  - 0: all other codes
- sbs_dbg_unlocked: unlock the debug (when set to 1) for the Cortex-M33 nonsecure.
  - 1: 0xB4
  - 0: all other codes
- sbs_dbg_unlocked_sec: unlock the debug (when set to 1) for the Cortex-M33 secure.
  - 1: 0xB4
  - 0: all other codes

14.3.8 SBS hardware secure storage control

This feature ensures the isolation of keys and data related to RoT (root-of-trust) when re-opening the debug (when product in the field).

This includes a dedicated area called OB-Keys in the flash memory (see *Section 7: Embedded flash memory (FLASH)* for more details), and the key derivation (DHUK) mechanism protecting data using a hardware key different for the identified domains.

Figure 90. SBS hardware secure storage control



- sbs_obk_hdpl: select secure storage context for current HDPL, or greater ones (to allow provisioning). This signal is set thanks to the SBS_NEXTHDPLCR register. This information is used by the flash memory to select the right area, and by the SAES when processing the derived key to generate a key derived related to HDPL context.
- sbs_epoch_out: EPOCH counters are used to manage the REPLAY protection. These counters are incremented when regressions are done. Injecting the EPOCH in the DHUK (in the SAES) ensures that all information encrypted with the DHUK cannot be replayed after a regression.
- sbs_sec_nsec: apply a different protection DHUK for secure and nonsecure assets.

All keys encrypted using the SAES/DHUK inherit of the RHUK property: unique per device.

All data encrypted thanks to the SAES using the DHUK are specific to a combination of [sbs_obk_hdpl + sbs_epoch_out + sbs_sec_nsec].

Inputs used to control the hardware secure storage control

sbs_epoch_in and sbs_epoch_in_ns: 24-bit values coming from the flash memory and representing regression counters respectively for secure and nonsecure.

Configuration

- NEXTHDPL in SBS_NEXTHDPLCR: access to next HDPL secure storage areas.
- EPOCH_SEL in SBS_EPOCHSELCR: select the EPOCH source related to the secure storage under control (SEC_EPOCH, NS_EPOCH, or FORCED_To_Zero).

Table 145. OBK-HDPL logic

HDPL[7:0] in SBS_HDPLSR	NEXTHDPL[1:0]			
	0x0	0x1	0x2	0x3
0 (0xB4)	0xB4	0x51	0x8A	0x6F
1 (0x51)	0x51	0x8A	0x6F	
2 (0x8A)	0x8A	0x6F		
3 (0x6F)	0x6F			
Others				

14.3.9 SBS flash banks control

Two registers control the flash bank swapping operation as shown in *Table 146*:

- FB_MODE value indicates the hardware physical address of flash banks.
- SWAP_LOCK value, when asserted, prevents any CPU access from modifying the bank mapping.

Table 146. Bank swap control

SWAP_LOCK	FB_MODE	Bank1 address	Bank2 address	FB_MODE change allowed	SWAP_LOCK change allowed
0xC3	0xC3	0x0800_0000	0x0804_00000	0xB4	0xB4
0xC3	0xB4	0x0804_0000	0x0800_0000	0xC3	
0xB4	0xC3	0x0800_0000	0x0804_0000	no change allowed	
0xBA	0xB4	0x0804_0000	0x0800_0000		

Both FB_MODE and SWAP_LOCK access permissions are restricted to secure state if TrustZone is enabled (TZEN = 0xB4), and always restricted to privileged mode whatever the TrustZone is enabled or disabled. Any illegal access is reported in TZIC.

Hardware initialization

- FB_MODE inherits after reset from the value of the SWAP_BANK option byte.
- SWAP_LOCK inherits after reset from the value of the SWAP_LOCKOUT option byte.

14.4 SBS interrupts

SBS does not support interrupts.

14.5 SBS registers

14.5.1 SBS temporal isolation control register (SBS_HDPLCR)

Address offset: 0x010

Reset value: 0x0000 00B4

Reset: system reset

Register security: no restriction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR_HDPL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INCR_HDPL[7:0]**: increment HDPL value
0xB4: no increment
0x6A: recommended value to increment HDPL level by one
Others: all other values allow a HDPL level increment.

14.5.2 SBS temporal isolation status register (SBS_HDPLSR)

Address offset: 0x014

Reset value: 0xXXXX XXXX

The reset value depends on boot case: booting with HDPL0 for ST code, or HDPL1 for all other cases. See *Table 144* for more details.

Reset: system reset

Register security: no restriction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPL[7:0]**: temporal isolation level
This bitfield returns the current temporal isolation level.
0xB4: HDPL0, RSS
0x51: HDPL1, iRoT
0x8A: HDPL2, uRoT
0x6F: HDPL3, application (secure/nonsecure)

14.5.3 SBS next HDPL control register (SBS_NEXTHDPLCR)

Address offset: 0x018

Reset value: 0x0000 0000

Reset: system reset

Register security: no restriction

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NEXTHDPL[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **NEXTHDPL[1:0]**: index to point to a higher HDPL than the current one

Index to add to the current HDPL to point (through OBK-HDPL) to the next secure storage areas (OBK-HDPL = HDPL + NEXTHDPL). See *Table 145* for more details.

14.5.4 SBS debug control register (SBS_DBGCR)

Address offset: 0x020

Reset value: 0x0000 0000

Reset: debug reset (system reset or power-on reset)

Register security: HDPL0/1, RAZ/WI otherwise

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_AUTH_SEC[7:0]								DBG_AUTH_HDPL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_UNLOCK[7:0]								AP_UNLOCK[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DBG_AUTH_SEC[7:0]**: control debug opening secure/nonsecure

Write 0xB4 to this bitfield to open debug for secure and nonsecure.
Writing any other values only open nonsecure.

Bits 23:16 **DBG_AUTH_HDPL[7:0]**: authenticated debug temporal isolation level

Writing to this bitfield defines at which HDPL the authenticated debug opens.
0x51: HDPL1
0x8A: HDPL2
0x6F: HDPL3

Note: Writing any other values is ignored. Reading any other value means the debug never opens.

Bits 15:8 **DBG_UNLOCK[7:0]**: debug unlock when DBG_AUTH_HDPL is reached

Write 0xB4 to this bitfield to open the debug when HDPL in SBS_HDPLSR equals to DBG_AUTH_HDPL in this register.

Bits 7:0 **AP_UNLOCK[7:0]**: access port unlock
Write 0xB4 to this bitfield to open the device access port.

14.5.5 SBS debug lock register (SBS_DBGLOCKR)

Address offset: 0x024

Reset value: 0x0000 00B4

Reset: debug reset (system reset or power-on reset)

Register security: HDPL0/1, RAZ/WI otherwise

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCFG_LOCK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DBGCFG_LOCK[7:0]**: debug configuration lock
Reading this bitfield returns 0x6A if the bitfield value is different from 0xB4.
0xC3 is the recommended value to lock the debug configuration using this bitfield.
0xB4: Writes to SBS_DBGCR and SBS_DBGCR2 allowed (default)
Others: Writes to SBS_DBGCR and SBS_DBGCR2 ignored

14.5.6 SBS debug control register 2 (SBS_DBGCR2)

Address offset: 0x028

Reset value: 0x0000 0000

Reset: debug reset (system reset or power-on reset)

Register security: HDPL0/1, RAZ/WI otherwise

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_AP_UNLOCK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SEC_AP_UNLOCK[7:0]**: Secure AHB access port unlock.
Write 0xB4 to this bitfield to open the secure AHB access port.

14.5.7 SBS RSS command register (SBS_RSSCMDR)

Address offset: 0x034

Reset value: 0x0000 0000

Reset: power-on reset

Register security: always secure (RAZ/WI if nonsecure)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSSCMD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RSSCMD[15:0]**: RSS command

The application can use this bitfield to pass on a command to the RSS, executed at the next reset.

When RSSCMD ≠ 0 and PRODUCT_STATE is in Open, then the system always boots on RSS whatever is the boot pin value.

14.5.8 SBS EPOCH selection control register (SBS_EPOCHSELCR)

Address offset: 0x0A0

Reset value: 0x0000 0000

Reset: system reset

Register security: Secure when TZ_STATE = 1 (RAZ, WI in nonsecure access). Nonsecure protection when TZ_STATE = 0. This register is protected by privileged whatever is TZ_STATE, RAZ/WI if non privileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH_SEL[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **EPOCH_SEL[1:0]**: select EPOCH value to be sent to the SAES

00: NS_EPOCH (nonsecure) counter input selected

01: SEC_EPOCH counter input selected

1x: EPOCH forced to zero

14.5.9 SBS security mode configuration control register (SBS_SECCFGR)

Address offset: 0x0C0

Reset value: 0x0000 0000

Reset: system reset

Register security: always secure. RAZ/WI if nonsecure transaction and TZ_STATE = 1. RAZ/WI if TZ_STATE = 0.

This register is programmed by secure software if the user wants functions configurable through system registers to be secure or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSEC	Res.	CLASSBSEC	SBSSEC
												rw		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FPUSEC**: FPU security enable
0: SBS_FPUIMR register accessible through secure or nonsecure transaction
1: SBS_FPUIMR register accessible only through secure transaction
Note: This bit can be written only through privilege transaction.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CLASSBSEC**: ClassB security enable
0: SBS_CFGR2 register accessible through secure or nonsecure transaction
1: SBS_CFGR2 register accessible only through secure transaction

Bit 0 **SBSSEC**: SBS clock control, memory-erase status register and compensation cell register security enable
0: SBS_MESR, SBS_CCCSR, SBS_CCVALR, SBS_CCSWCR registers accessible through secure or nonsecure transaction
1: SBS_MESR, SBS_CCCSR, SBS_CCVALR, SBS_CCSWCR registers accessible only through secure transaction

14.5.10 SBS product mode and configuration register (SBS_PMCR)

Address offset: 0x100

Reset value: 0x0000 0000

Reset: system reset

Register security:

- When TrustZone is activated (TZEN = 0xB4):
  - depending on I/O PBx_SEC_EN secure input for PBx_FMP configuration bits
- When TrustZone is disabled (TZEN = 0xC3), there is no access restriction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB9_FMP	Res.	PB7_FMP	PB6_FMP
												rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C4SMB3EN	I2C3SMB3EN	Res.	I2C1SMB3EN
												rw	rw		rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **PB9_FMP**: Fast-mode Plus driving capability activation on PB9

This bit can be read and written only with secure access if PB9 is secure in GPIOB. This bit enables the Fm+ driving mode for PB9 when PB9 is not used by I2C peripheral. This can be used to drive a LED for instance.

0: PB9 pin operates in standard mode.

1: Fm+ mode is enabled on PB9 pin and the speed control is bypassed.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PB7_FMP**: Fast-mode Plus driving capability activation on PB7

This bit can be read and written only with secure access if PB7 is secure in GPIOB. This bit enables the Fm+ driving mode for PB7 when PB7 is not used by I2C peripheral. This can be used to drive a LED for instance.

0: PB7 pin operates in standard mode.

1: Fm+ mode is enabled on PB7 pin and the speed control is bypassed.

Bit 16 **PB6_FMP**: Fast-mode Plus driving capability activation on PB6

This bit can be read and written only with secure access if PB6 is secure in GPIOB. This bit enables the Fm+ driving mode for PB6 when PB6 is not used by I2C peripheral. This can be used to drive a LED for instance.

0: PB6 pin operates in standard mode.

1: Fm+ mode is enabled on PB6 pin and the speed control is bypassed.

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **I2C4SMB3EN**: SMBUS 3.0 on I2C4 enable

0: SMBUS 3.3 on I2C4 disable

1: SMBUS 3.3 on I2C4 enable

Note: Depending on device

Bit 2 **I2C3SMB3EN**: SMBUS 3.3 on I2C3 enable
0: SMBUS 3.3 on I2C3 disable
1: SMBUS 3.3 on I2C3 enable
Note: Depending on device

Bit 1 Reserved, must be kept at reset value.

Bit 0 **I2C1SMB3EN**: SMBUS 3.3 on I2C1 enable
0: SMBUS 3.3 on I2C1 disable
1: SMBUS 3.3 on I2C1 enable

14.5.11 SBS FPU interrupt mask register (SBS_FPUIMR)

Address offset: 0x104

Reset value: 0x0000 001F

Reset: system reset

Register security: depends on FPUSEC in SBS_SECCFGR

This register is accessible only through privilege transaction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **FPU_IE[5:0]**: FPU interrupt enable
Set and cleared by software to enable the Cortex-M33 FPU interrupts
FPU_IE[5]: inexact interrupt enable (interrupt disabled at reset)
FPU_IE[4]: input abnormal interrupt enable
FPU_IE[3]: overflow interrupt enable
FPU_IE[2]: underflow interrupt enable
FPU_IE[1]: divide-by-zero interrupt enable
FPU_IE[0]: invalid operation interrupt enable

14.5.12 SBS memory erase status register (SBS_MESR)

Address offset: 0x108

Reset value: 0x0000 000X (bit 0 is not affected by system reset)

Register security: depends on SBSSEC in SBS_SECCFGR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
															rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IPMEE**: ICACHE erase status
This bit is set by hardware when ICACHE and PKA RAM erase is completed after potential and confirmed tamper detection or a product state regression. Refer to *Section 50: Tamper and backup registers (TAMP)* for more details.
This bit is cleared by software by writing 1 to it.
0: ICACHE and PKA RAM erase on going
1: ICACHE and PKA RAM erase done

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MCLR**: device memories erase status
This bit is set by hardware when SRAM1, SRAM2, SRAM3, ICACHE, DCACHE and PKA RAM erase is completed after power-on reset or tamper detection or product state regression. Refer to *Section 50: Tamper and backup registers (TAMP)*.
This bit is not reset by system reset and is cleared by software by writing 1 to it.
0: memory erase on going if not yet cleared by software
1: Memory erase done

14.5.13 SBS compensation cell for I/Os control and status register (SBS_CCCSR)

Address offset: 0x110

Reset value: 0x0000 0000

Reset: system reset

Register security: depends on SBSSEC in SBS_SECCFGR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY1	Res.	Res.	Res.	Res.	Res.	Res.	CS1	EN1
							r							rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RDY1**: VDD compensation cell ready flag
This bit provides the status of the compensation cell.
0: VDD compensation cell not ready
1: VDD compensation cell ready (code value provided by the cell can be used)

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **CS1**: code selection for VDD power rail (reset value set to 1)
This bit selects the code to be applied for the I/O compensation cell.
0: Code from the cell (available in the SBS_CCVALR)
1: Code from SBS_CCSWCR

Bit 0 **EN**1: enable compensation cell for VDD power rail
This bit enables the I/O compensation cell.
0: I/O compensation cell disabled
1: I/O compensation cell enabled

14.5.14 SBS compensation cell for I/Os value register (SBS_CCVALR)

Address offset: 0x114

Reset value: 0x0000 0088

Reset: system reset

Register security: depends on SBSSEC in SBS_SECCFGR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APSRC1[3:0]				ANSRC1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **APSRC1[3:0]**: compensation value for the PMOS transistor

This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

Bits 3:0 **ANSRC1[3:0]**: compensation value for the NMOS transistor

This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

14.5.15 SBS compensation cell for I/Os software code register (SBS_CCSWCR)

Address offset: 0x118

Reset value: 0x0000 0078

Reset: system reset

Register security: depends on SBSSEC in SBS_SECCFGR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_APSRC1[3:0]				SW_ANSRC1[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SW_APSRC1[3:0]**: PMOS compensation code for the VDD power rails

This bitfield is written by software to define an I/O compensation cell code for PMOS transistors of the VDD power rail. This code is applied to the I/O when CS1 is set in SBS_CCCSR.

Bits 3:0 **SW_ANSRC1[3:0]**: NMOS compensation code for VDD power rails

This bitfield is written by software to define an I/O compensation cell code for NMOS transistors of the VDD power rail. This code is applied to the I/O when CS1 is set in SBS_CCCSR.

14.5.16 SBS Class B register (SBS_CFGR2)

Address offset: 0x120

Reset value: 0x0000 0000

Reset: system reset

Register security: depends on CLASSBSEC in SBS_SECCFGR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCECCL	ECCL	PVDL	SEL	CLL
											rs	rw	rs	rs	rs

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DCECCL**: Arm® Cortex®-M33 D-CACHE ECC lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Arm® Cortex®-M33 D-CACHE double ECC error with break input of TIM1/8/15/16/17.

0: Arm® Cortex®-M33 D-CACHE double ECC error flag disconnected to timer break inputs

1: Arm® Cortex®-M33 D-CACHE double ECC error flag connected to timer break inputs

Bit 3 **ECCL**: ECC lock

This bit is set and cleared by software. It can be used to enable and lock the flash memory double ECC error with break input of TIM1/8/15/16/17.

0: double ECC error flag disconnected to timer break inputs

1: double ECC error flag connected to timer break inputs

Bit 2 **PVDL**: PVD lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection with TIM1/8/15/16/17 break inputs.

0: PVD interrupt disconnected from timer break inputs. PVDE and PLS[2:0] in the PWR registers are read/write

1: PVD interrupt is connected to timer break inputs. PVDE and PLS[2:0] in the PWR registers are read only

Bit 1 **SEL**: SRAM ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM double ECC error signal with break input of TIM1/8/15/16/17.

0: SRAM double ECC error flag disconnected from timer break inputs

1: SRAM double ECC error flag connected to timer break inputs

Bit 0 **CLL**: core lockup lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the lockup (HardFault) output of Cortex-M33 with TIM1/8/15/16/17 break inputs.

0: lockup output disconnected from timer break inputs

1: lockup output connected to timer break inputs

14.5.17 SBS CPU nonsecure lock register (SBS_CNSLCKR)

Address offset: 0x144

Reset value: 0x0000 0000

Reset: system reset

Register security: This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

This register is used to lock the configuration of nonsecure MPU and VTOR_NS registers of the Cortex-M33.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LOCKNSMPU**: nonsecure MPU register lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to nonsecure MPU_CTRL_NS, MPU_RNR_NS and MPU_RBAR_NS registers.

0: nonsecure MPU registers write enabled

1: nonsecure MPU registers write disabled

Bit 0 **LOCKNSVTOR**: VTOR_NS register lock

This bit is set by software and cleared only by a system reset.

0: VTOR_NS register write enabled

1: VTOR_NS register write disabled

14.5.18 SBS CPU secure lock register (SBS_CSLCKR)

Address offset: 0x148

Reset value: 0x0000 0000

Reset: system reset

Register security: This register can be written only when the access is secure/privilege.

A nonsecure read/write access is RAZ/WI and generates an illegal access event. When the system is not secure (TZ_STATE = 0), this register is RAZ/WI.

This register is used to lock the configuration of PRIS and BFHFNMINS in the AIRCR, SAU, secure MPU and VTOR_S registers of the Cortex-M33.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
													rs	rs	rs

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKSAU**: SAU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers.

0: SAU registers write enabled

1: SAU registers write disabled

Bit 1 **LOCKSMPU**: secure MPU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to secure MPU_CTRL, MPU_RNR and MPU_RBAR registers.

0: Secure MPU registers writes enabled

1: Secure MPU registers writes disabled

Bit 0 **LOCKSVTAIRCR**: VTOR_S and AIRCR register lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to VTOR_S register, PRIS and BFHFNMINS bits in the AIRCR register.

0: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write enabled

1: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write disabled

14.5.19 SBS ECC NMI mask register (SBS_ECCNMIR)

Address offset: 0x14C

Reset value: 0x0000 0000

Reset: system reset

Register security: secure access only when TZ_STATE = 1 (RAZ/WI in nonsecure access). No security protection if TZ_STATE = 0.

This register is accessible only through privilege transaction.

This register sets up the expected behavior on NMI regarding double ECC errors from the flash memory.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI_MASK_EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ECCNMI_MASK_EN**: NMI behavior setup when a double ECC error occurs on FLASH data part

0: NMI generated if a double ECC error in the FLASH data part

1: NMI not generated if a double ECC error in the FLASH data part

14.5.20 SBS debug control register 2 (SBS_SWAPLOCKR)

Address offset: 0x170

Reset value: 0x0000 00B4

Reset: debug reset (system reset or power-on reset)

Register security:

- Secure access only when TZ_STATE = 1 (RAZ/WI in nonsecure access). No security protection if TZ_STATE = 0.
- This register is accessible only through privilege transaction.
- Write ignored if SBS_SWAPLOCKR.SWAP_LOCK = 0xB4.

This register is used to prevent specific configurations on memory remap.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_LOCK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SWAP_LOCK[7:0]**: Flash bank swap operation lock

0xC3: Flash bank swap operation allowed, FB_MODE and SWAP_BANK option byte can be modified
0xB4: Flash bank swap operation not allowed, FB_MODE and SWAP_BANK option byte access blocked

14.5.21 SBS debug control register 2 (SBS_FBMODER)

Address offset: 0x174

Reset value: 0x0000 00C3

Reset: debug reset (system reset or power-on reset)

Register security:

- Secure access only when TZ_STATE = 1 (RAZ/WI in nonsecure access). No security protection if TZ_STATE = 0.
- This register is accessible only through privilege transaction.
- Write ignored if SBS_SWAPLOCKR.SWAP_LOCK = 0xB4.

This register is used for specific configurations on memory remap.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FB_MODE[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FB_MODE[7:0]**: Flash bank mode selection

0xC3: Flash Bank 1 mapped at 0x0800 0000 (and aliased @0x0C00 0000) and Flash Bank 2 mapped at 0x0804 0000 (and aliased at 0x0C04 0000)
0xB4: Flash Bank 2 mapped at 0x0800 0000 (and aliased @0x0C00 0000) and Flash Bank 1 mapped at 0x0804 0000 (and aliased at 0x0C04 0000)

14.5.22 SBS register map

Table 147. SBS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SBS_HDPLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR_HDPL[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x014	SBS_HDPLSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
	Reset value																									x	x	x	x	x	x	x	x

Table 147. SBS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	SBS_ NEXTHDPLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NEXTHDPL[1:0]	
	Reset value																															0	0
0x020	SBS_DBGCR	DBG_AUTH_SEC[7:0]								DBG_AUTH_HDPL[7:0]								DBG_UNLOCK[7:0]								AP_UNLOCK[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SBS_DBGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCFG_LOCK[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x028	SBS_DBGCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_AP_UNCLOCK[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C to 0x030	Reserved	Reserved																															
0x034	SBS_RSSCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSCMD[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x09C	Reserved	Reserved																															
0x0A0	SBS_ EPOCHSELCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH_SEL[1:0]	
	Reset value																															0	0
0x0A4 to 0x0BC	Reserved	Reserved																															
0x0C0	SBS_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSEC	Res.	CLASSBSEC	SBSSEC
	Reset value																													0		0	0
0x0C4 to 0x0FC	Reserved	Reserved																															
0x100	SBS_PMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB9_FMP	Res.	PB7_FMP	PB6_FMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C4SMB3EN	I2C3SMB3EN	Res.	I2C1SMB3EN
	Reset value													0		0	0													0	0		0
0x104	SBS_FPUIMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
	Reset value																											1	1	1	1	1	1
0x108	SBS_MESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
	Reset value																0																X
0x110	SBS_CCCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY1	Res.	Res.	Res.	Res.	Res.	Res.	CS1	EN1
	Reset value																								0							0	0
0x114	SBS_CCVALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APSRC2 [3:0]				ANSRC2 [3:0]				APSRC1 [3:0]				ANSRC1 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0

Table 147. SBS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x118	SBS_CCSWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_APSRC2 [3:0]				SW_ANSRC2 [3:0]				SW_APSRC1 [3:0]				SW_ANSRC1 [3:0]			
	Reset value																	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0
0x120	SBS_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCECCL	ECCL	PVDL	SEL	CLL
	Reset value																												0	0	0	0	0
0x124 to 0x140	Reserved	Reserved																															
0x144	SBS_CNSLCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
	Reset value																															0	0
0x148	SBS_CSLCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
	Reset value																														0	0	0
0x14C	SBS_ECCNMIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI_MASK_EN
	Reset value																																0
0x170	SBS_SWAPLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_LOCK[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x174	SBS_FBMODER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FB_MODE[7:0]							
	Reset value																									1	1	0	0	0	0	1	1

Refer to *Section 2.3* for the register boundary addresses.

15 Peripherals interconnect matrix

15.1 Interconnect matrix introduction

Several peripherals have direct connections between them, enabling autonomous communication and/or synchronization: this approach saves CPU resources and power supply consumption. In addition, these hardware connections remove software latency and help the design of a predictable system.

Depending on peripherals, these interconnections can operate in Run, Sleep, and Stop modes.

15.2 Connection summary

Table 148. Peripherals interconnect matrix[1][2][3]

Source	Destination																															
	HRTIM1	TIM1	TIM8	TIM20	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM26	TIM27	TIM9	TIM10	TIM11	TIM12	TIM13	TIM14	TIM15	TIM16	TIM17	LPTIM1	ADC1/2/3	COMP1..8	DAC1/2/5..10	OPAMP1/2/3	GPDMA1/2	TAMP	RTC	AES/SAES/PKA	CRS	PLAY1	EXTI	POSENC
HRTIM 1	1	1	1	1	1	1	1	-	-	-	-	-	1	1	1	1	1	1	1	-	-	1	2	-	4	-	-	-	-	-	-	16	-	-
TIM1	1	-	1	1	1	1	1	1	-	-	-	-	1	1	1	1	1	1	1	1	1	-	2	20	4	-	-	-	-	-	-	16	-	23
TIM8	1	1	-	1	1	1	1	1	-	-	-	-	1	1	1	1	1	1	1	1	1	-	2	20	4	-	-	-	-	-	-	16	-	23
TIM20	1	1	1	-	1	1	1	1	-	-	-	-	1	1	1	1	1	1	1	1	1	-	2	20	4	-	-	-	-	-	-	16	-	23
TIM2	1	1	1	1	-	1	1	1	-	-	-	-	1	1	1	1	1	1	1	-	-	-	2	20	4	-	10	-	-	-	-	16	-	-
TIM3	1	1	1	1	1	-	1	1	-	-	-	-	1	1	1	1	1	1	1	-	-	-	2	20	-	-	-	-	-	-	-	16	-	-
TIM4	1	1	1	1	1	1	-	1	-	-	-	-	1	1	1	1	1	1	1	-	-	-	2	20	4	-	-	-	-	-	-	16	-	-
TIM5	1	1	1	1	1	1	1	-	-	-	-	-	1	1	1	1	1	1	1	-	-	-	-	-	4	-	10	-	-	-	-	-	-	-
TIM6	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	4	-	10	-	-	-	-	16	-	-
TIM7	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	10	-	-	-	-	-	-	-
TIM26	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	4	-	-	-	-	-	-	-	-	-
TIM27	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	4	-	-	-	-	-	-	-	-	-
TIM9	1	1	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	20	4	-	10	-	-	-	-	16	-	-
TIM10	1	1	1	1	1	1	1	1	-	-	-	-	1	-	-	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	16	-	-
TIM11	1	1	1	1	1	1	1	1	-	-	-	-	1	-	-	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM12	1	1	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	20	4	-	10	-	-	-	-	16	-	-

Table 148. Peripherals interconnect matrix[1][2][3] (continued)

Source	Destination																																		
	HRTIM1	TIM1	TIM8	TIM20	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM26	TIM27	TIM9	TIM10	TIM11	TIM12	TIM13	TIM14	TIM15	TIM16	TIM17	LPTIM1	ADC1/2/3	COMP1..8	DAC1/2/5..10	OPAMP1/2/3	GPDMA1/2	TAMP	RTC	AES/SAES/PKA	CRS	PLAY1	EXTI	POSENC
TIM13	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	-	*1*	-	-	*1*	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM14	-	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	-	*1*	-	-	*1*	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM15	*1*	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	-	-	*1*	-	-	*1*	-	-	-	*1*	*1*	-	*2*	*20*	*4*	-	*10*	-	-	-	-	*16*	-	-
TIM16	-	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	-	-	*1*	-	-	*1*	-	-	*1*	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-
TIM17	-	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	-	-	*1*	-	-	*1*	-	-	*1*	*1*	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*2*	-	*4*	-	*10*	-	-	-	-	*16*	*9*	-
I3C1	-	-	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	*5*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*9*	-
ADC1	*3*	*3*	-	-	*3*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-
ADC2	*3*	-	*3*	-	-	*3*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	-	-	*16*	-	-
ADC3	*3*	-	*3*	*3*	-	-	*3*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	-	-	*16*	-	-
GPDMA1/2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*10*	-	-	-	-	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*6*	*2*	-	*4*	-	*10*	-	-	-	-	-	-	-
RTC wake-up	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*7*	-	-	-	-	-	-	*10*	-	-	-	-	-	*9*	-
RTC alarm	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*6*	-	-	-	-	*10*	*13*	-	-	-	-	*9*	-

Table 148. Peripherals interconnect matrix[1][2][3] (continued)

Source	Destination																																	
	HRTIM1	TIM1	TIM8	TIM20	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM26	TIM27	TIM9	TIM10	TIM11	TIM12	TIM13	TIM14	TIM15	TIM16	TIM17	LPTIM1	ADC1/2/3	COMP1..8	DAC1/2/5..10	OPAMP1/2/3	GPDMA1/2	TAMP	RTC	AES/SAES/PKA	CRS	PLAY1	EXTI	POSENC
TAMP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*6*	-	-	-	-	*10*	-	*14*	*15*	-	-	*9*	-
HSI	-	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	*5*	-	-	*5*	-	-	*5*	-	-	-	-	-	-	-	-	-	-	-	-
HSE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	-	-	-	-	-	-	-	-	-	-	-
LSE	-	-	-	-	*5*	-	-	-	-	-	-	-	-	-	-	-	-	-	*5*	*5*	-	*5*	-	-	*12*	-	-	-	-	-	*19*	-	*9*	-
LSI	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*5*	-	*5*	-	-	-	-	-	-	-	-	-	-	-	-
CSS in LSE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	-	-	-	-	-
CSS in HSE	*8*	*8*	*8*	*8*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*8*	*8*	*8*	-	-	-	-	-	-	*13*	-	-	-	-	-	-
MCO1	-	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	*5*	-	-	*5*	-	*5*	-	-	-	-	-	-	-	-	-	-	*16*	-	-
MCO2	-	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	*5*	-	-	*5*	-	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-
VCORE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*11*	-	-	-	-	-	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*11*	*22*	-	-	-	-	-	-	-	-	-	-
Vsense	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*11*	-	-	-	-	-	-	-	-	-	-	-
VDD / Temp. monitor	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	-	-	-	-	-

Table 148. Peripherals interconnect matrix[1][2][3] (continued)

Source	Destination																																
	HRTIM1	TIM1	TIM8	TIM20	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM26	TIM27	TIM9	TIM10	TIM11	TIM12	TIM13	TIM14	TIM15	TIM16	TIM17	LPTIM1	ADC1/2/3	COMP1..8	DAC1/2/5..10	OPAMP1/2/3	GPDMA1/2	TAMP	RTC	AES/SAES/PKA	CRS	PLAY1	EXTI	POSENC
System errors	*8*	*8*	*8*	*8*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*8*	*8*	*8*	-	-	-	-	-	-	-	-	-	-	*16*	-	-
System flash memory	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*15*	-	-	-	-
AES/SAES/PKA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	*15*	-	-	-	-
RNG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	-	-	-	-	-
USB	-	-	-	-	*17*	*17*	*17*	*17*	-	-	-	-	*17*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*19*	-	*9*	-
PLAY1	*16*	*16*	*16*	*16*	*16*	*16*	*16*	-	-	-	-	-	*16*	*16*	*16*	*16*	*16*	*16*	*16*	*16*	*16*	*16*	*16*	*16*	*16*	-	*16*	-	-	-	-	-	*16*	-
OPAMP 1/2/3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*11*	*22*	-	-	-	-	-	-	-	-	-	-
COMP1..8	*18*	*18*	*18*	*18*	*18*	*18*	*18*	-	*18*	-	-	-	*18*	*18*	*18*	*18*	*18*	*18*	*18*	*18*	*18*	*6, 18*	-	-	-	-	-	-	-	-	-	*16*	*9*	-
DAC1/2/5..10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*22*	-	*21*	-	-	-	-	-	-	-	-

1. Numbers in this table are links to corresponding subsections of *Section 15.3*.
2. "-" means no interconnect.
3. Refer to the product datasheet for the availability of peripherals. If the peripheral is not present, consider the associated interconnection as unavailable.

15.3 Interconnection details

15.3.1 Master to slave interconnection for timers

From timer (TIM1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/20/26/27 and HRTIM1) to timer (TIM1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/20/26/27 and HRTIM1).

Purpose

Some timers are linked together internally for synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

A description of the feature is provided in *Section 42.4.27: Timer synchronization*.

The synchronization modes are detailed in:

- *Section 41* for advanced-control timers TIM1/8/20
- *Section 42* for general-purpose timers TIM2/3/4/5
- *Section 44* for the general-purpose timers TIM9/TIM10/TIM11/TIM12/TIM13/TIM14
- *Section 45* for the general-purpose timer TIM15/TIM16/TIM17
- *Table 411: External events mapping and associated features* for HRTIM1
- *Table 412: Update enable inputs and sources* for HRTIM1
- *Table 413: Burst mode clock sources* for HRTIM1
- *Table 414: Burst mode trigger source* for HRTIM1
- *Table 415: Internal synchronization source* for HRTIM1

Triggering signals

The output (from master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1/8/20) following a configurable timer event. It can be also from signals timx_oc1 in case of TIM10/11/13/14/16/17. The input (to slave) is on signals timx_itry.

The possible master/slave connections are given in:

- *Table 451* for advanced-control timers TIM1/8/20
- *Table 477* for general-purpose timers TIM2/3/4/5
- *Table 503* for the general-purpose timers TIM9/TIM10/TIM11/TIM12/TIM13/TIM14
- *Table 515* for the general-purpose timer TIM15/TIM16/TIM17

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 471: Effect of low-power modes on TIM1/TIM8/TIM20*
- *Table 492: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 508: Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14*
- *Table 529: Effect of low-power modes on TIM15/TIM16/TIM17*

15.3.2 Triggers to ADCs

From EXTI, timers (TIM1/2/3/4/6/7/8/9/12/15/16/17/20/26/27) and LPTIM1 and HRTIM1 to ADC1/ADC2/ADC3.

Purpose

A conversion (or a sequence of conversions) can be triggered either by software or by an external event (such as timer capture or input pins). For ADC1/ADC2/ADC3, if the EXTEN[1:0] (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, external events can trigger a conversion with the selected polarity.

More details in:

- *Table 226: ADC1/2/3 interconnection*
- EXTEN[1:0] defined in *ADC configuration register (ADC_CFGR1)*
- JEXTEN[1:0] defined in *ADC injected sequence register (ADC_JSQR)*

General-purpose timers (TIM2/3/4/5), basic timer (TIM6), advanced-control timers (TIM1/8/20), and general-purpose timer (TIM9/12/15/16/17) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo.

Low-power timers (LPTIM1) can be used to generate the ADC triggering event through the LPTIM channels, in addition to the EXTI on channels 11 and 15 (for ADC1/2) and EXTI on channels 6 and 7 (for ADC3).

HRTIM timer can be used to generate the ADC triggering event through the HRTIM output trigger signals.

Triggering signals

For ADC1/ADC2/ADC3, the input triggering signals and the description of the interconnection between ADC1/ADC2/ADC3 and timers are given in:

- adc_ext_trg: *Table 226: ADC1/2/3 interconnection*
- adc_jext_trg: *Table 226: ADC1/2/3 interconnection*
- *Section 26.4.21: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*

Active power mode

This interconnection is active in Run and Sleep modes for all ADCs. The timers are active only in Run and Sleep modes. The effects of low-power modes are given in:

- *Table 471: Effect of low-power modes on TIM1/TIM8/TIM20*
- *Table 492: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 529: Effect of low-power modes on TIM15/TIM16/TIM17*
- *Table 499: Effect of low-power modes on TIM6/TIM7/TIM26/TIM27*
- *Table 508: Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14*
- *Table 535: STM32G56x/55x/54x/53x LPTIM features*
- *Table 546: Effect of low-power modes on the LPTIM*

15.3.3 ADC analog watchdogs as triggers to timers

From ADC1/ADC2/ADC3 to TIM1/8/20, TIM2/3/4 and HRTIM1.

Purpose

The internal analog watchdog output signals coming from ADC1/ADC2/ADC3 are connected to on-chip timers and HRTIM. ADC1/ADC2/ADC3 can provide trigger events through analog watchdog signals to advanced-control timers (TIM1/8/20) and general-

purpose timers (TIM2/TIM3/TIM4) to reset, start, stop, or clock the counter. ADC1/ADC2/ADC3 can provide trigger events through analog watchdog signals to HRTIM1 external event signals for various functions in HRTIM.

Settings description of the ADC analog watchdog and timer trigger, are provided in:

- *Section 41.3.6: External trigger input* for TIM1/8/20
- *Section 42.4.5: External trigger input* for TIM2/3/4
- *Table 457: Interconnect to the tim_etr input multiplexer* for TIM1/8/20
- *Table 483: Interconnect to the tim_etr input multiplexer* for TIM2/3/4
- *Section 15.3.12: Internal analog signals to ADCs* for the ADC1/ADC2/ADC3/ADC_AWDy_OUT signal output generation
- *Table 411: External events mapping and associated features* for HRTIM1

Triggering signals

The output (from ADC) is on signals ADCn_AWDx_OUT, with n being the ADC instance and x = 1, 2, 3 (three watchdogs per ADC). The input (to timer) is on signal TIMx_ETR (external trigger). The input (to HRTIM) is on signal hrtim_eevy (external event).

Active power mode

ADC1/ADC2/ADC3 are active in Run and Sleep modes.

15.3.4 From comparators to timers (TIMx, HRTIM, LPTIM)

From COMP (COMP1 to 8) to timer (TIM1/2/3/4/5/8/910/11/12/13/14/15/16/17/20 and LPTIM1).

Purpose

The comparators (COMP1..8) output can be used as timers (TIM1/2/3/4/5/8/910/11/12/13/14/15/16/17/20, LPTIM1) input (tim_ti_in, tim_itr, tim_etr, tim_ocref_clr, tim_brk_cmp, tim_brk2_cmp) for various purposes (triggers, clearing, break input). The comparators (COMP1..8) output can be used as HRTIM (HRTIM1) external event input or fault input (hrtim_eev, hrtim_eev).

Triggering signals

Comparators outputs selection to timers and HRTIM inputs is provided in:

- *Table 478: Interconnect to the tim_ti1 input multiplexer* for TIM2/TIM3/TIM4/TIM5
- *Table 479: Interconnect to the tim_ti2 input multiplexer* for TIM2/TIM3/TIM4/TIM5
- *Table 483: Interconnect to the tim_etr input multiplexer* for TIM2/TIM3/TIM4/TIM5
- *Table 484: Interconnect to the tim_ocref_clr input multiplexer* for TIM2/TIM3/TIM4/TIM5
- *Table 516: Interconnect to the tim_ti1 input multiplexer* for TIM15/TIM16/TIM17
- *Table 519: Timer break interconnect* for TIM15/TIM16/TIM17
- *Table 521: Interconnect to the ocref_clr input multiplexer* for TIM15/TIM16/TIM17
- *Table 504: Interconnect to the tim_ti1 input multiplexer* for TIM9/TIM10/TIM11/TIM12/TIM13/TIM14
- *Table 452: Interconnect to the tim_ti1 input multiplexer* for TIM1/TIM8/TIM20
- *Table 457: Interconnect to the tim_etr input multiplexer* for TIM1/TIM8/TIM20
- *Table 458: Interconnect to the tim_brk_cmp multiplexer* for TIM1/TIM8/TIM20
- *Table 306: Timer break2 interconnect* for TIM1/TIM8/TIM20
- *Table 460: Interconnect to tim_ocref_clr multiplexer* for TIM1/TIM8/TIM20
- *Section 46.4.3: LPTIM input and trigger mapping* for LPTIM1
- *Table 411: External events mapping and associated features* for HRTIM1
- *Table 416: Fault inputs* for HRTIM1

Active power mode

Interconnect to timers is active in Run and Sleep modes.

Comparators and LPTIM1 are active in Run, Sleep, and Stop modes.

- *Table 546: Effect of low-power modes on the LPTIM*
- *Table 264: Comparator behavior in low-power modes*

15.3.5 Triggers to DAC

From timer (TIM1/2/3/4/5/6/7/8/9/15/20/26/27), HRTIM, low-power timers (LPTIM1), and EXTI to DAC.

Purpose

General-purpose timers (TIM2/3/4/5/8/9/15), basic timers (TIM6/7/26/27), advanced-control timers (TIM1/8/20), LP timers (LPTIM1) outputs channels (lptim1_ch1), HRTIM1, and EXTI can be used as triggering event to start a DAC conversion.

Triggering signals

The output (from timer) on the TIMx_TRGO signal from LPTIM1 and from HRTIM1 timers are directly connected to corresponding DAC inputs.

The selection of input triggers on DAC is provided in:

- *Table 250: DAC interconnection*
- *Section 28.4.7: DAC trigger selection*
- *Table 418: HRTIM DAC triggers connections*

Active power mode

This interconnect is active in Run and Sleep modes.

15.3.6 Clock sources to timers

From HSE, LSE, LSI, HSI, and MCO to timers (TIM2/9/12/15/16/17) and LP timers (LPTIM1).

Purpose

A timer input or counter can receive different clock sources, and can be used, for example, to calibrate the internal oscillator on a reference clock.

External clocks (HSE, LSE), internal clocks (LSI, HSI), and microcontroller output clock (MCO1, MCO2) can be used as input to timers:

- LSE is assigned to general purpose timer TIM2 as external inputs signals. LSE can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in). Inputs assignment and clock selection description are detailed in:
  - *Section 42.4.6: Clock selection* for TIM2
  - External clock mode1: *Table 504: Interconnect to the tim_ti1 input multiplexer* for TIM9/12, tim_ti1_in4 (HSI)
  - External clock mode2: *Table 483: Interconnect to the tim_etr input multiplexer* for TIM2, tim_etr22 (LSE)
- LSE, LSI, and HSE are assigned to general purpose timers TIM15/16/17 as external inputs signals. LSE/HSE can be selected as counter clock provided by an external clock source in mode1 (tim_ti1 or tim_ti2 signals). Inputs assignment and clock selection description are detailed in:
  - *Section 45.4.6: Clock selection* for TIM15/16/17. External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
  - *Table 516: Interconnect to the tim_ti1 input multiplexer* for TIM15/16/17, tim_ti1_in2, tim_ti1_in3, tim_ti1_in4, tim_ti1_in10
- Microcontroller output clock (MCO): MCO1 is connected as external input to general-purpose timer TIM15/17, MCO2 is connected as external input to general-purpose timer TIM15.
  - *Table 516: Interconnect to the tim_ti1 input multiplexer* for TIM15/16/17
- LSI and LSE can be selected as input capture 2 to LPTIM1 as described in *Table 540: LPTIM1 input 2 connections*.
- hsi_cal_ck (HSI/1024) can be selected as input capture 1 to LPTIM1 as described in *Table 539: LPTIM1 input 1 connections*.

Active power mode

This feature is available under Run and Sleep modes.

15.3.7 Triggers to low-power timers

From EXTI, TAMP, and RTC alarm to LP timers (LPTIM1).

Purpose

LPTIM1 counters can be started either by software, or after the detection of an active edge on one of the eight trigger inputs (see *Section 46.4.7: Trigger multiplexer*).

Triggering signals

This trigger feature is described in *Section 46.4.7: Trigger multiplexer* and the following sections. The input selection is described in *Table 538: LPTIM1 external trigger connections*.

Active power mode

This interconnection is active in Run, Sleep, and Stop modes.

15.3.8 RTC wake-up as inputs to timers

From RTC to timer (TIM16).

Purpose

RTC wake-up interrupt can be used as input to general-purpose timer (TIM16) channel 1.

Triggering signals

RTC wake-up signal is connected to tim_ti1_in4 signal as described in *Table 516: Interconnect to the tim_ti1 input multiplexer* for TIM16.

Active power mode

This interconnection is active down to Stop mode. Timers are not active but the count is performed at wake-up.

15.3.9 System errors as break signals to timers and HRTIM

From system errors to timers (TIM1/8/15/16/17/20).

Purpose

CSS, CPU lockup, SRAM1/2 ECC double errors, DCACHE ECC double-error detection, FLASH ECC double-error detection, and PVD can generate system errors in the form of timer break toward timers (TIM1/8/15/16/17/20) and HRTIM system fault input.

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

Triggering signals

The possible sources of break are described in:

- *Section 41.3.19: Using the break function* for TIM1/8/20
- *Section 45.4.17: Using the break function* for TIM15/16/17
- *Table 308: System break interconnect* for TIM1/8/20

- *Table 520: System break interconnect* for TIM15/16/17
- *Table 410: HRTIM inputs/outputs summary* for HRTIM

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 471: Effect of low-power modes on TIM1/TIM8/TIM20*
- *Table 529: Effect of low-power modes on TIM15/TIM16/TIM17*
- *Table 410: HRTIM inputs/outputs summary* for HRTIM

15.3.10 From timers to comparators

From timer (TIM1/2/3/4/8/9/12/15/20) to comparator (COMP1 to 8).

Purpose

Timers (TIM1/2/3/4/8/9/12/15/20) can generate blanking window for comparators (COMP1 to 8).

Triggering signals

The timers outputs (tim_oc) selection to comparator blanking signal (comp_blk) is provided in:

- *Table 263: COMPx blanking source assignment (COMP1 to COMP8)*

Active power mode

This interconnect is active in Run and Sleep modes.

15.3.11 Triggers to GPDMA1/2

From EXTI, RTC (alarm/wake-up), TAMP, timers (TIM2/5/6/7/9/12/15), LP timers (LPTIM1), GPDMA transfer complete (gpdma1_chx_tc, gpdma2_chx_tcf) to GPDMA1/2.

Purpose

A GPDMA trigger can be assigned to GPDMA channel x. A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event. The trigger mode can also be programmed to condition the LLI link transfer. More details are given in the sections below:

- *Section 16.3.7: GPDMA triggers*
- *Section 16.4.13: GPDMA triggered transfer*
- *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details on:
  - Trigger selection TRIGSEL[5:0] field
  - Trigger mode (LLI) defined by TRIGM[1:0]
  - Trigger polarity as defined by TRIGPOL[1:0]

Triggering signals

GPDMA trigger mapping is specified in *Table 155: Programmed GPDMA1/2 trigger*, according to GPDMA_CxTR2.TRIGSEL[5:0].

Active power mode

This interconnection remains functional in Sleep mode.

Refer to:

- *Section 16.6: GPDMA in low-power modes*

15.3.12 Internal analog signals to ADCs

From internal analog source to ADC (ADC1/2/3).

Purpose

The internal reference voltage (V_{REFINT}), the internal temperature sensor (V_{SENSE}), the internal digital core voltage (V_{DDCORE}), and the OPAMPs internal outputs (opamp1_int, opamp2_int, opamp3_int) are connected to ADC (ADC1/2/3) input channels.

This is according to:

- *Table 226: ADC1/2/3 interconnection*
- *Section 26.4.12: Channel selection (ADC_SQRy, ADC_JSQR)*

Active power mode

This interconnect is active in Run and Sleep modes.

15.3.13 Internal analog signals to COMPs

From internal analog sources to COMP (COMP1 to 8).

Purpose

The internal reference voltage (1/4 V_{REFINT}, 1/2 V_{REFINT}, 3/4 V_{REFINT}, V_{REFINT}), the DAC outputs (DAC1/2/5/6/7/8/9/10) are connected to the COMP (COMP1 to 8) input channels.

- *Table 262: COMPx_INM inverting input assignment (COMP1 to COMP8)*

The internal OPAMP (OPAMP1/2/3) are connected to the COMP (COMP1 to 8) input channels.

- *Table 261: COMPx_INP non-inverting input assignment (COMP1 to COMP8)*

Active power mode

This feature remains available in Run, Sleep, and Stop modes.

15.3.14 Internal analog signals to OPAMPs

From internal analog sources to OPAMP (OPAMP1/3).

Purpose

The DAC outputs (DAC5/6) are connected to the OPAMP (OPAMP1/3) input channels.

- *Table 267: OPAMP pin and signal assignment per instance*

Active power mode

This feature remains available in Run, Sleep, and Stop modes.

15.3.15 Clock source for the DAC sample and hold mode

From LSI/LSE to DACs.

Purpose

DACs can run in Stop mode. The sample and hold block and its associated registers use the LSI or LSE clock source (dac_hold_ck) in Stop mode.

Table 249: DAC internal input/output signals: dac_hold_ck

Active power mode

This feature remains available in Run, Sleep, and Stop modes.

15.3.16 Internal tamper sources

From internal peripherals, clocks, or monitoring, to tamper.

Purpose

To detect any abnormal activity or tentative to corrupt the device, embedded tampers alert the system of undesired events. Different actions can be taken as consequence.

The list of tamper sources can be found in *Table 570: TAMP interconnection*.

Active power mode

This interconnection is active in all power modes if the tamper source is activated.

15.3.17 Output from tamper to RTC

From TAMP to RTC.

Purpose

The RTC can timestamp a tamper event to retrieve history of the detection. The RTC can also control GPIOs, and set a signal based on tamp or alarm status outside the MCU.

Refer to section *Section 49.3.18: Timestamp function* and *Section 49.6.24: RTC timestamp on tamper control register (RTC_TAMPTSCR)* for more details.

Active power mode

This interconnection remains active in all power modes.

15.3.18 PLAY inputs/outputs from/to internal peripherals

PLAY1 inputs and outputs connection to on-chip peripherals.

Purpose

PLAY1 peripheral can implement additional function as response from many inputs from internal peripherals and the control of peripherals by providing output to peripherals.

Input/output signals selection into/from PLAY1 are provided in:

- *Table 202: PLAY1 logic array input n signal selection*
- *Table 203: PLAY1 output connections*

Active power mode

This feature remains available in Run, Sleep, and Stop modes.

15.3.19 Encryption keys to AES/SAES

From TAMP backup registers, system flash memory to and in between SAES and AES.

Purpose

The encryption mechanism requires a hardware key that must be stored in a protected nonvolatile memory. Different approaches are implemented to load them in a non-readable way. Tamper backup registers or system flash memory can be used to store respectively BHK or RHUK, and to implement a dedicated bus to pass it to the SAES.

Refer to *Section 35.4.14: SAES operation with wrapped keys* for more details.

The AES encryption mechanism (faster than the SAES) can benefit from the sharing key of the SAES. Refer to *Section 35.4.15: SAES operation with shared keys* for more details.

Active power mode

AES and SAES are operating under Run and Sleep modes.

15.3.20 From USB to timers

From USB_SOF signal to TIM2/3/4/5/9.

Purpose

The USB signal usb_sof can be used as trigger for timers.

Triggering signals

Description of the USB source for timer trigger, are provided in:

- *Table 483: Interconnect to the tim_etr input multiplexer* for TIM2/3/4/5
- *Table 504: Interconnect to the tim_ti1 input multiplexer* for TIM9.

Active power mode

This feature is available under Run and Sleep modes.

15.3.21 CRS clock sources

From various clock signals to clock recovery system (CRS).

Purpose

Precise clock signal can be used as synchronization clock for clock recovery system (CRS).

Triggering signals

Description of the clock sources for CRS are provided in:

- *Table 135: CRS interconnection*

Active power mode

This feature is available under Run and Sleep modes.

15.3.22 Internal sources for EXTI lines

From various peripherals outputs to EXTI lines.

Purpose

Some peripherals able to generate wake-up or interrupt events for the input to EXTI lines.

Triggering signals

Description of the EXTI lines are provided in:

- *Table 176: EXTI line connections*

Active power mode

This feature is available under Run, Sleep and Stop modes.

15.3.23 Internal signals to POSENC

Purpose

Synchronization signal from timers (TIM1/8/20) to POSENC1/2 strobe interface to synchronize position reading.

Triggering signals

Description of the POSENC strobe signals are provided in:

- *Section 57.3.7: POSENC strobe interface*

Active power mode

This feature is available under Run and Sleep modes.

16 General purpose direct memory access controller (GPDMA)

This section applies only to STM32G55x/56x devices.

16.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

16.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half-transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 8 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 16.3.2*)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level and byte-based padding

  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers
  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 16.3.2*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Programmable GPDMA request and trigger selection
  - Programmable GPDMA half-transfer and transfer complete events generation
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers
  - Channel abort and restart
  - Channel terminate with FIFO draining
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and nonsecure GPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any GPDMA secure resource (register, register field) from a nonsecure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

16.3 GPDMA implementation

16.3.1 GPDMA instances

There are two instances of the GPDMA in the devices, named as GPDMA1 and GPDMA2.

Each GPDMA instance has the same channel-based implementation and is connected to the same requests and triggers, as detailed in the following sub-sections.

16.3.2 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

Table 149. GPDMA1/2 channel implementation

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 3	2	0	Channel x (x = 0 to 3) is implemented with: – a FIFO of 8 bytes, 2 words – fixed/contiguously incremented addressing These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and SRAM.
x = 4 to 5	4	0	Channel x (x = 4 to 5) is implemented with: – a FIFO of 32 bytes, 8 words – fixed/contiguously incremented addressing These channels may be used for GPDMA transfers between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.
x = 6 to 7	4	1	Channel x (x = 6 to 7) is implemented with: – a FIFO of 32 bytes, 8 words – 2D addressing These channels may be used for GPDMA transfers between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.

16.3.3 GPDMA in low-power modes

The GPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 150. GPDMA1/2 wake-up in low-power modes

Feature	Low-power modes
Wake-up	GPDMA1/2 in Sleep mode

16.3.4 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[7:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 151. Programmed GPDMA1/2 request

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
0	adc1_dma
1	adc2_dma
2	dac1_ch1_dma
3	dac2_ch1_dma
4	tim6_upd_dma

Table 151. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
5	tim7_upd_dma
6	spi1_rx_dma
7	spi1_tx_dma
8	spi2_rx_dma
9	spi2_tx_dma
10	spi3_rx_dma
11	spi3_tx_dma
12	i2c1_rx_dma
13	i2c1_tx_dma
14	Reserved
15	i2c2_rx_dma
16	i2c2_tx_dma
17	Reserved
18	i2c3_rx_dma
19	i2c3_tx_dma
20	Reserved
21	usart1_rx_dma
22	usart1_tx_dma
23	usart2_rx_dma
24	usart2_tx_dma
25	usart3_rx_dma
26	usart3_tx_dma
27	uart4_rx_dma
28	uart4_tx_dma
29	uart5_rx_dma
30	uart5_tx_dma
31	uart6_rx_dma
32	uart6_tx_dma
33	Reserved
34	Reserved
35	Reserved
36	Reserved
37	hrtim1_master_dma
38	hrtim1_timA_dma
39	hrtim1_timB_dma

Table 151. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
40	hrtim1_timC_dma
41	hrtim1_timD_dma
42	hrtim1_timE_dma
43	hrtim1_timF_dma
44	Reserved
45	lpuart1_rx_dma
46	lpuart1_tx_dma
47	Reserved
48	Reserved
49	Reserved
50	Reserved
51	Reserved
52	Reserved
53	adc3_dma
54	Reserved
55	Reserved
56	Reserved
57	xspi1_dma
58	tim1_cc1_dma
59	tim1_cc2_dma
60	tim1_cc3_dma
61	tim1_cc4_dma
62	tim1_upd_dma
63	tim1_trgi_dma
64	tim1_com_dma
65	tim8_cc1_dma
66	tim8_cc2_dma
67	tim8_cc3_dma
68	tim8_cc4_dma
69	tim8_upd_dma
70	tim8_trgi_dma
71	tim8_com_dma
72	tim2_cc1_dma
73	tim2_cc2_dma
74	tim2_cc3_dma

Table 151. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
75	tim2_cc4_dma
76	tim2_upd_dma
77	tim2_trgi_dma
78	tim3_cc1_dma
79	tim3_cc2_dma
80	tim3_cc3_dma
81	tim3_cc4_dma
82	tim3_upd_dma
83	tim3_trgi_dma
84	tim4_cc1_dma
85	tim4_cc2_dma
86	tim4_cc3_dma
87	tim4_cc4_dma
88	tim4_upd_dma
89	tim4_trgi_dma
90	tim5_cc1_dma
91	tim5_cc2_dma
92	tim5_cc3_dma
93	tim5_cc4_dma
94	tim5_upd_dma
95	tim5_trgi_dma
96	tim15_cc1_dma
97	tim15_cc2_dma
98	tim15_upd_dma
99	tim15_trgi_dma
100	tim15_com_dma
101	tim16_cc1_dma
102	tim16_upd_dma
103	tim17_cc1_dma
104	tim17_upd_dma
105	lptim1_ic1_dma
106	lptim1_ic2_dma
107	lptim1_ue_dma
108	Reserved
109	Reserved

Table 151. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
110	Reserved
111	aes_out_dma
112	aes_in_dma
113	hash1_in_dma
114	ucpd1_rx_dma
115	ucpd1_tx_dma
116	cordic1_rd_dma
117	cordic1_wr_dma
118	fmac1_rd_dma
119	fmac1_wr_dma
120	saes_out_dma
121	saes_in_dma
122	i3c1_rx_dma
123	i3c1_tx_dma
124	i3c1_tc_dma
125	i3c1_rs_dma
126	i2c4_rx_dma
127	i2c4_tx_dma
128	fmac2_rd_dma
129	fmac2_wr_dma
130	Reserved
131	Reserved
132	Reserved
133	Reserved
134	Reserved
135	dac5_ch1_dma
136	dac6_ch1_dma
137	dac7_ch1_dma
138	dac8_ch1_dma
139	dac9_ch1_dma
140	dac10_ch1_dma
141	tim20_cc1_dma
142	tim20_cc2_dma
143	tim20_cc3_dma
144	tim20_cc4_dma

Table 151. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
145	tim20_upd_dma
146	tim20_trgi_dma
147	tim20_com_dma
148	Reserved
149	Reserved
150	Reserved
151	Reserved
152	Reserved
153	Reserved
154	Reserved
155	tim26_upd_dma
156	tim27_upd_dma

16.3.5 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 152. Programmed GPDMA1/2 request as a block request

GPDMA block requests
lptim1_ue_dma

16.3.6 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 153. GPDMA1/2 channel with peripheral early termination

GPDMA channel x with peripheral early termination
x = 0

This GPDMA support is activated when the channel x is programmed with GPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before the GPDMA has transferred the whole block (see *Section 16.4.15* for more details).

Table 154. Programmed GPDMA1/2 request with peripheral early termination

Programmed GPDMA channel x request with peripheral early termination
i3c1_rx_dma

16.3.7 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[5:0] in GPDMA_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 155. Programmed GPDMA1/2 trigger

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6
7	exti7
8	tamp_trgi1
9	tamp_trgi2
10	Reserved
11	lptim1_ch1
12	lptim1_ch2
13	Reserved
14	Reserved
15	rtc_alra_trgi
16	rtc_alrb_trgi
17	rtc_wut_trgi
18	gpdma1_ch0_tc
19	gpdma1_ch1_tc
20	gpdma1_ch2_tc
21	gpdma1_ch3_tc
22	gpdma1_ch4_tc
23	gpdma1_ch5_tc
24	gpdma1_ch6_tc
25	gpdma1_ch7_tc
26	gpdma2_ch0_tc
27	gpdma2_ch1_tc
28	gpdma2_ch2_tc
29	gpdma2_ch3_tc
30	gpdma2_ch4_tc

Table 155. Programmed GPDMA1/2 trigger (continued)

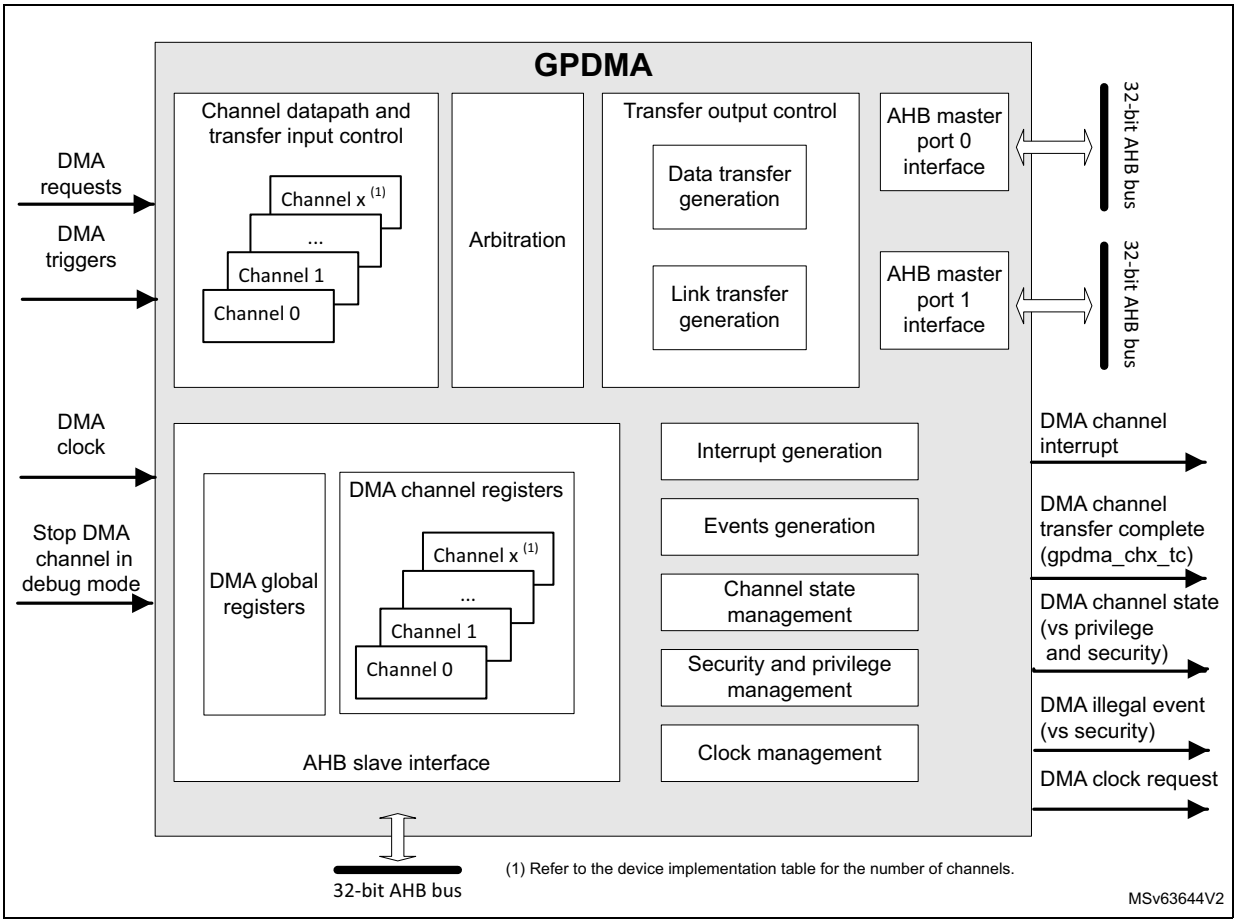
GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
31	gpdma2_ch5_tc
32	gpdma2_ch6_tc
33	gpdma2_ch7_tc
34	tim2_tgo
35	tim15_tgo
36	tim12_tgo
37	play1_out3/play1_out15(1)
38	tim9_tgo
39	tim7_tgo
40	tim6_tgo

1. play1_out3 for GPDMA1, play1_out15 for GPDMA2

16.4 GPDMA functional description

16.4.1 GPDMA block diagram

Figure 91. GPDMA block diagram



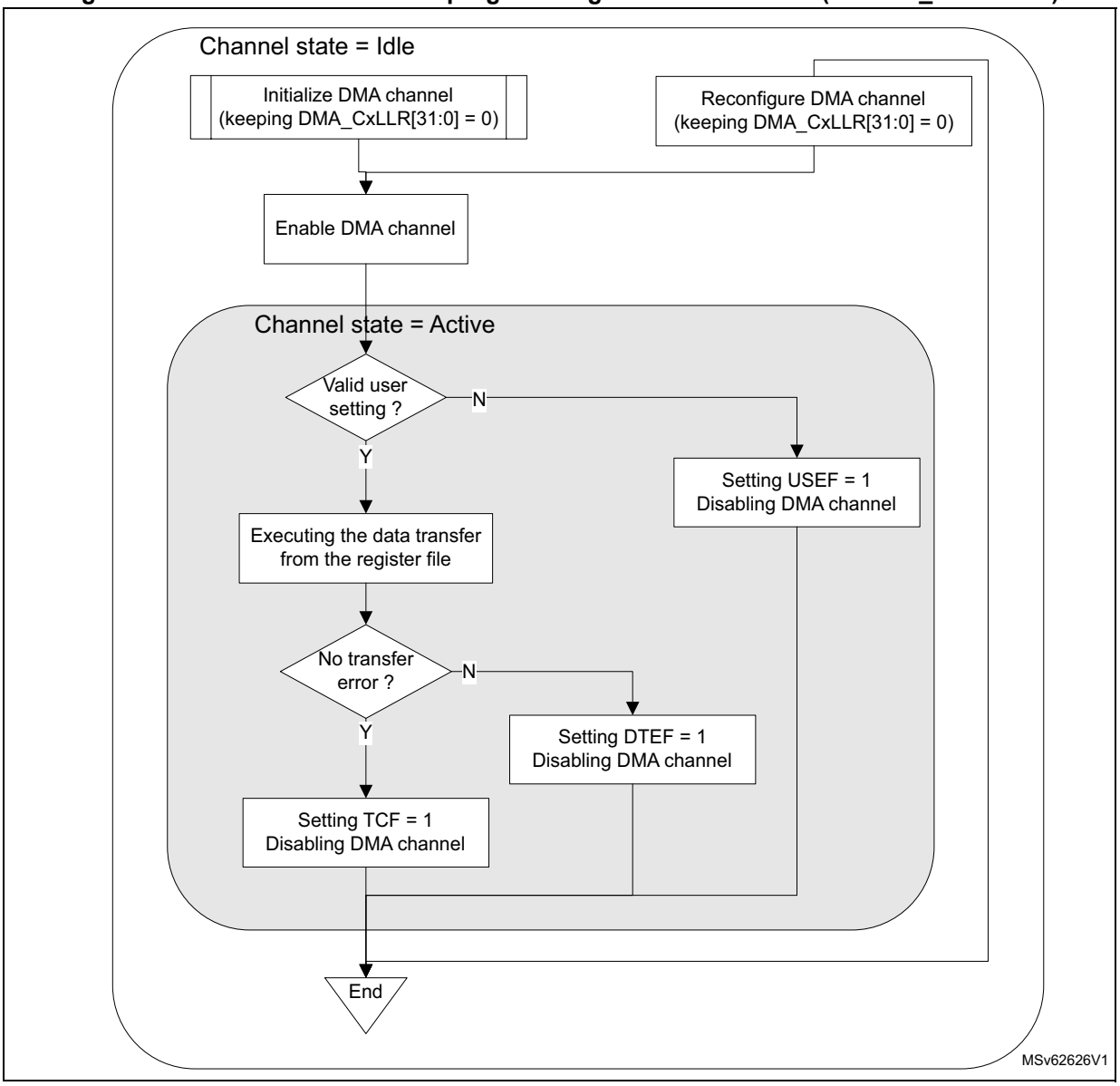
16.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

Figure 92 illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 92. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



16.4.3 GPDMA channel suspend and resume

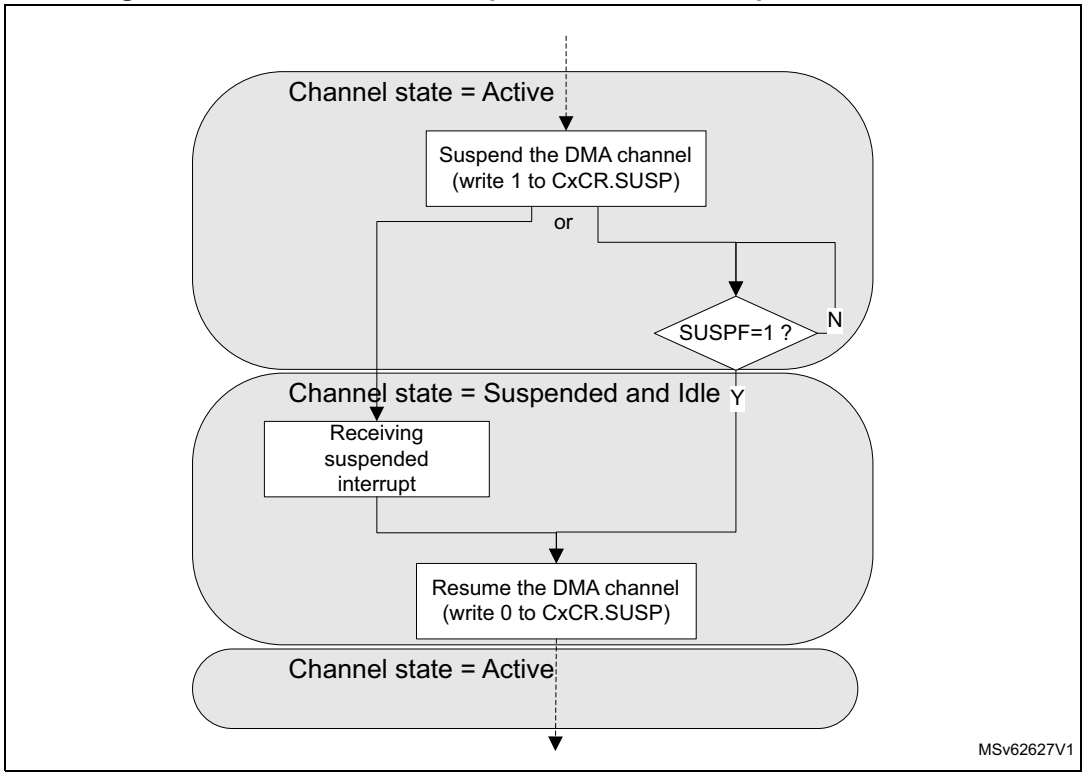
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note: An ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

Figure 93. GPDMA channel suspend and resume sequence



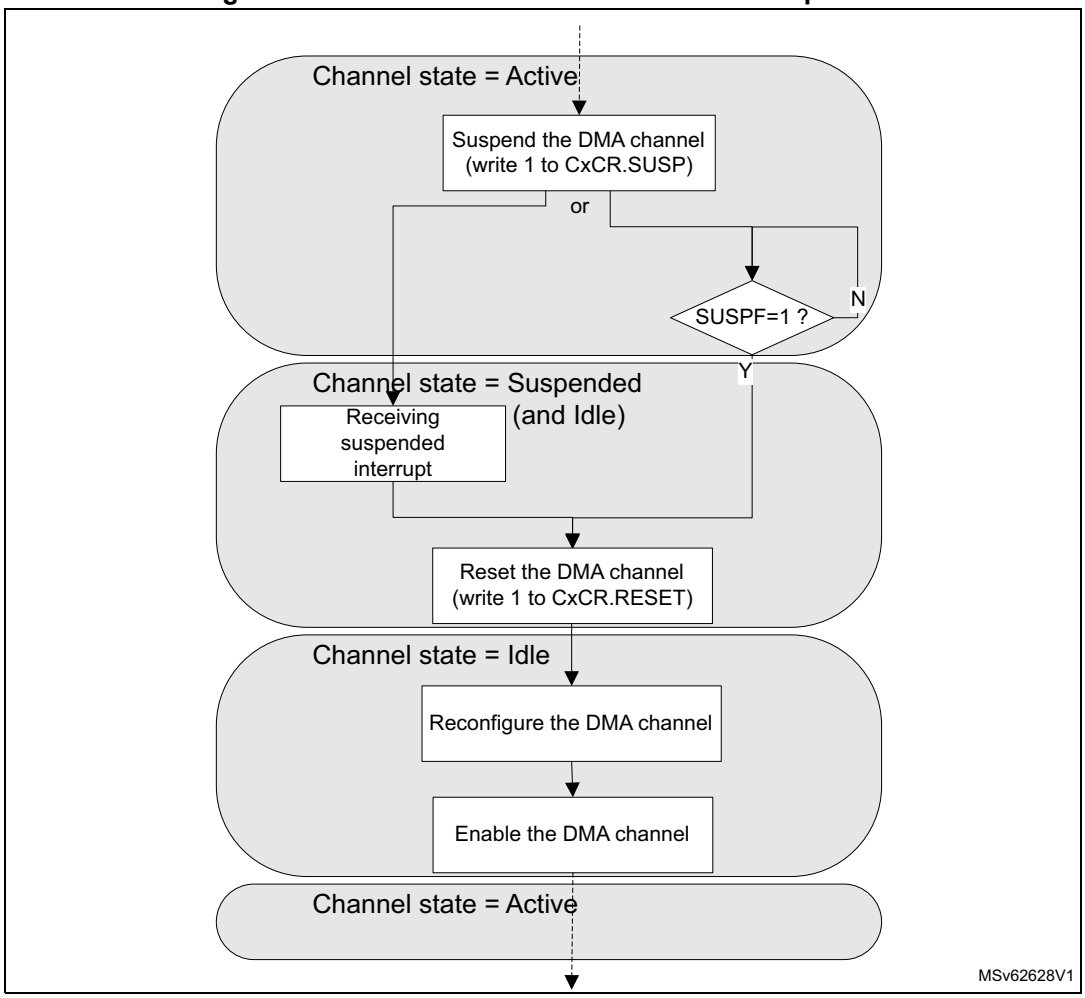
Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

16.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

Figure 94. GPDMA channel abort and restart sequence

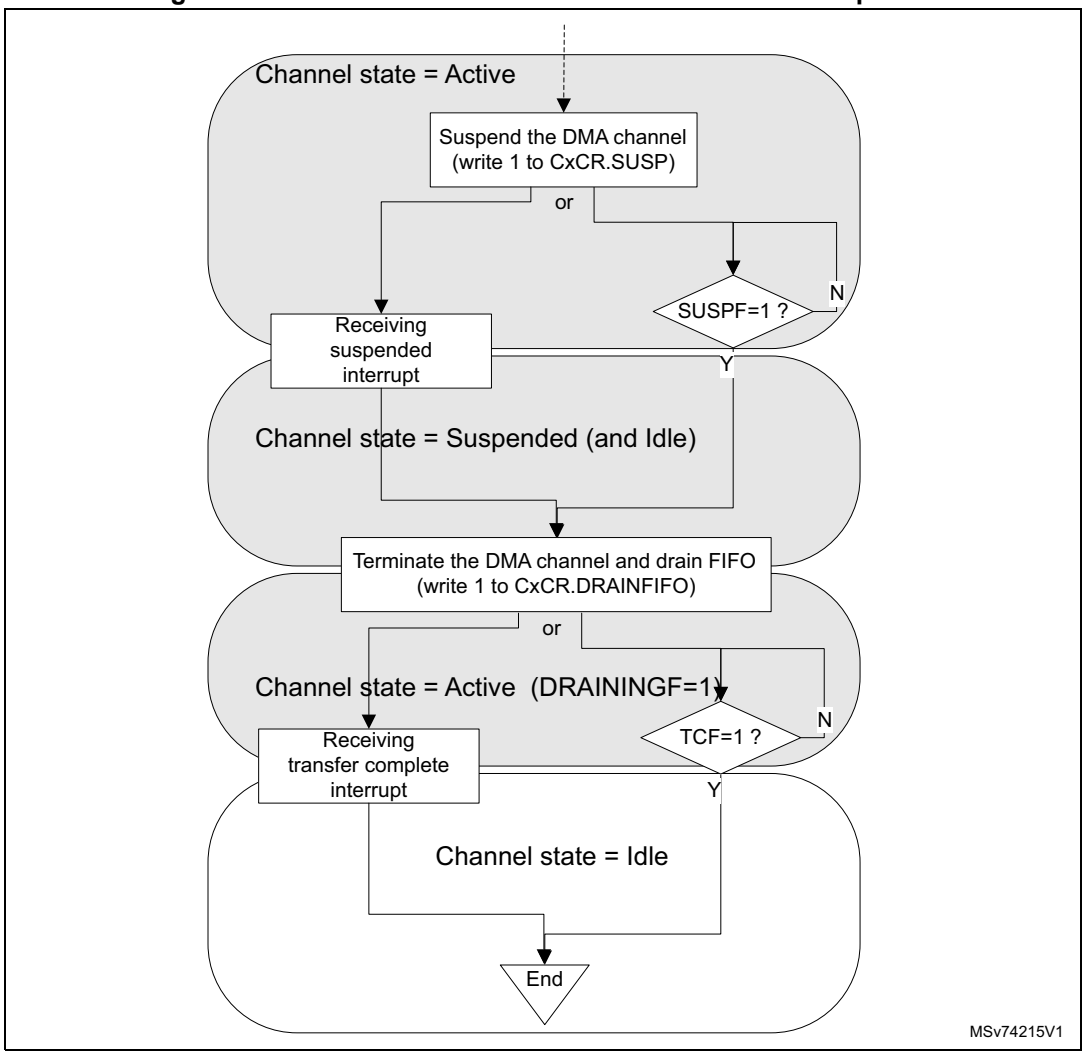


16.4.5 GPDMA channel terminate and drain FIFO

Alternatively, for terminating a continuous GPDMA transfer with a circular buffering or a double buffering, and having popped out any residual byte(s) in the channel FIFO and writing them to the destination, the software can terminate an active channel and drain the channel FIFO with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software orders to drain the channel FIFO by writing 1 to GPDMA_CxCR.DRAINFIFO.
4. The software polls transfer complete flag GPDMA_CxSR.TCF until TCF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.TCIE. Wait for the channel to be effectively with a drained FIFO means wait for completion of any residual write burst from the FIFO to the destination. Then the channel is disabled by hardware and in IDLE state.

Figure 95. GPDMA channel terminate and drain FIFO sequence



Note: If the softwares writes the GPDMA_CxCR register in a single shot with concurrently SUSP = 1, DRAINFIFO = 1, and RESET = 0, then the sequence for terminating and draining the FIFO also occurs.

GPDMA_CxTR2.DREQ = 1 and GPDMA_CxCR.DRAINFIFO = 1 is not supported, only peripheral-to-memory and memory-to-memory transfers are supported to be terminated with drained FIFO.

16.4.6 GPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the following UT1, UT2, UB1, USA, UDA, UT3 and UB2 if present, and ULL bits in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, GPDMA_CxTR3 and GPDMA_CxBR2 if present, and GPDMA_CxLLR) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Caution: The user must program the pointer to the next linked-list data structure (GPDMA_CxLLR[15:0]) not to exceed the 64-Kbyte addressable space defined by the link base address register (GPDMA_CxLBAR). The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA, plus UB2 and UT3 if present, and ULL) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 5) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 96*) and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory
- channel x (x = 6 to 7), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, GPDMA_CxTR3, GPDMA_CxBR2 and GPDMA_CxLLR, (see *Figure 97*), and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory

Figure 96. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

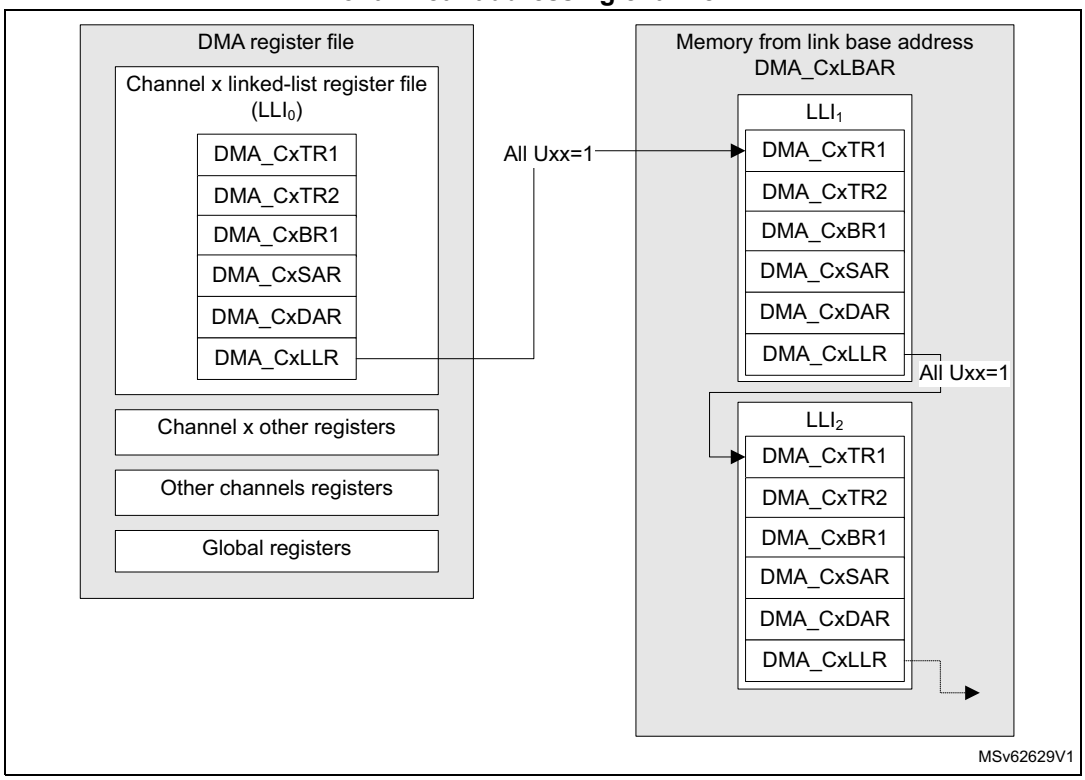
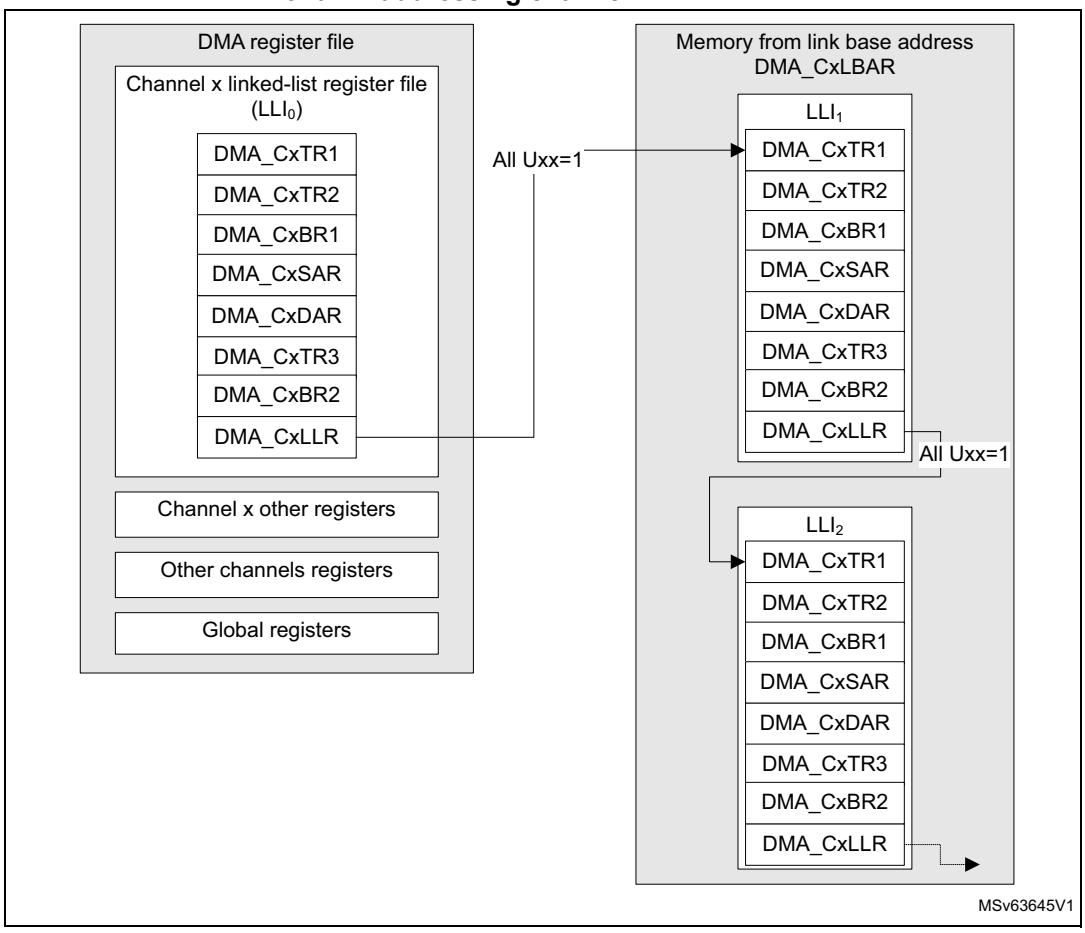


Figure 97. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see *Figure 98*).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in *Figure 99*.

Figure 98. GPDMA dynamic linked-list data structure of a linear addressing channel x

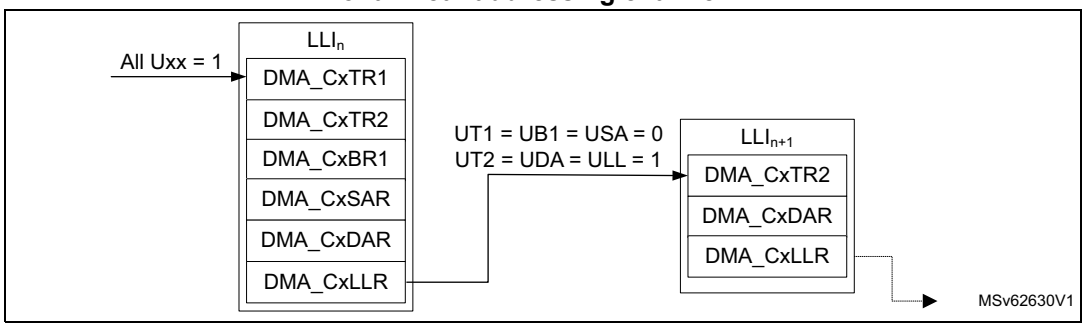
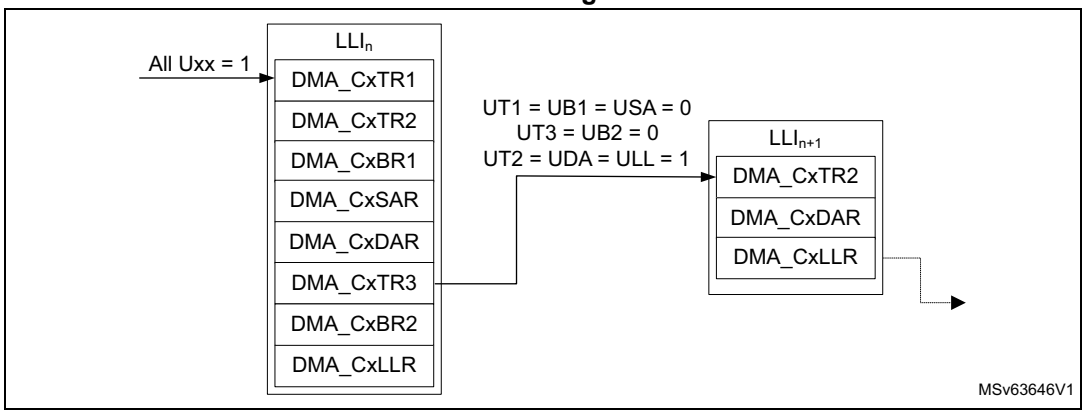


Figure 99. GPDMA dynamic linked-list data structure of a 2D addressing channel x



16.4.7 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the GPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.

16.4.8 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started

by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated with the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 16.4.15* for more details).

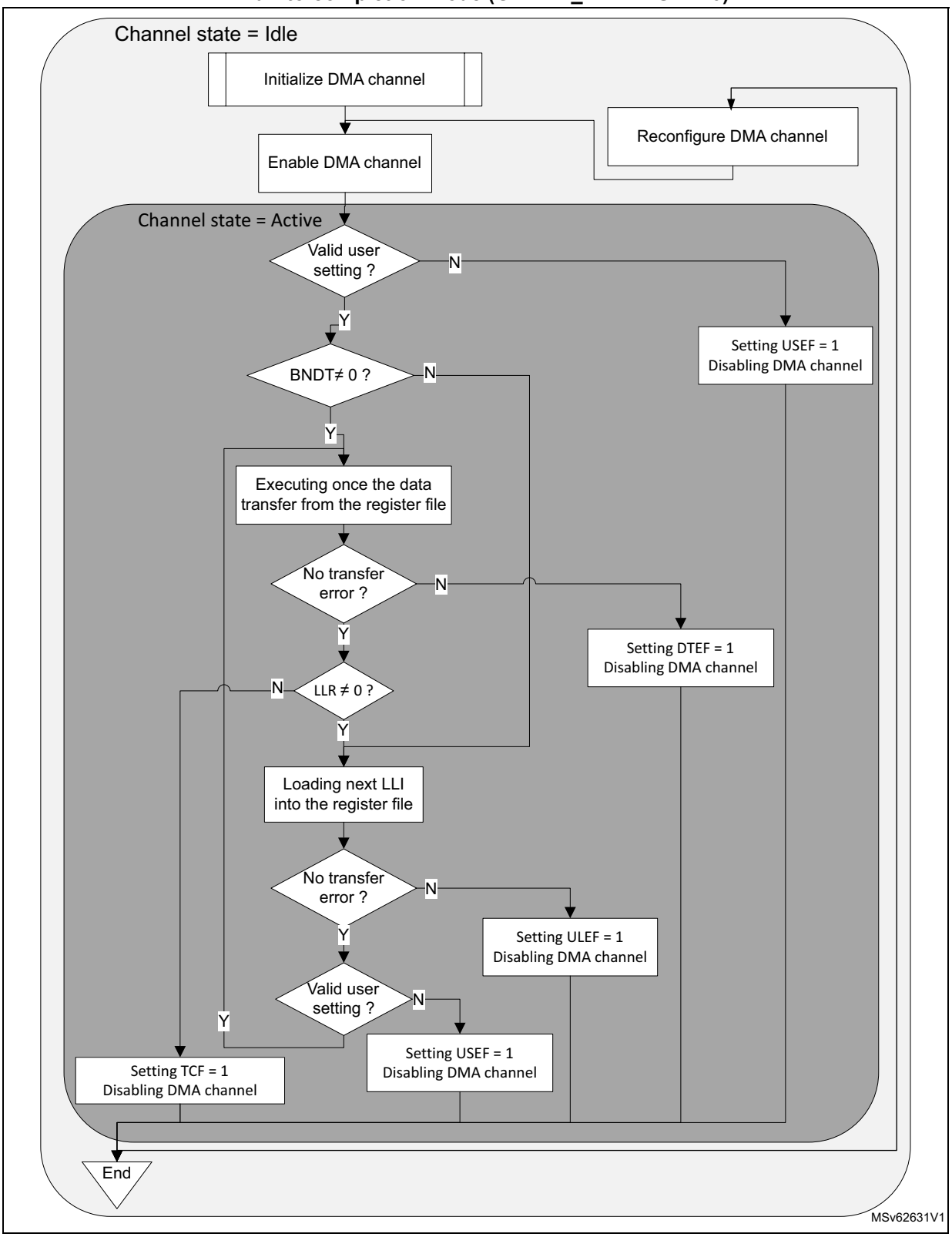
In typical run-to-completion mode, allocation of an GPDMA channel, including detailed programming, occurs once during GPDMA initialization. This process reserves a data communication link and GPDMA service during run-time for repeated transfers, such as transfers from a peripheral to memory, from memory to a peripheral, or memory-to-memory transfers. The reserved data communication link may consist of a dedicated channel, or the channel may be shared. A repeated transfer consists of a sequence of linked list items (LLIs).

Figure 100 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 100 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 100, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).

Figure 100. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

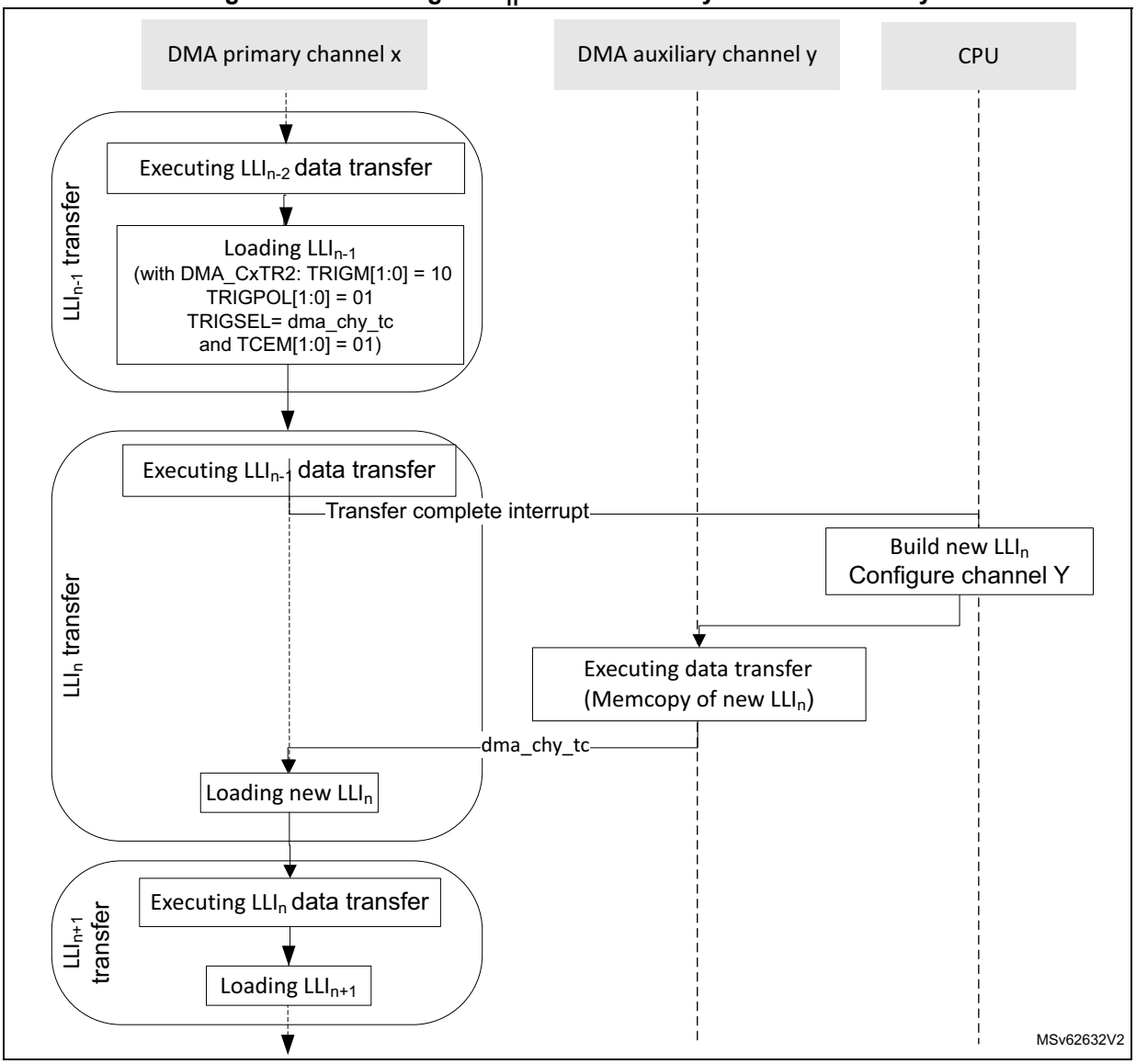
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 16.3.7* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 101 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 101. Inserting a LLI_n with an auxiliary GPDMA channel y



16.4.9 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 6 to 7), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.

  - GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
  - GPDMA_CxTR3 for channel x (x = 6 to 7) defines the source/destination additional address offset between burst transfers.
  - GPDMA_CxBR2 for channel x (x = 6 to 7) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated with the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 16.4.15* for more details).

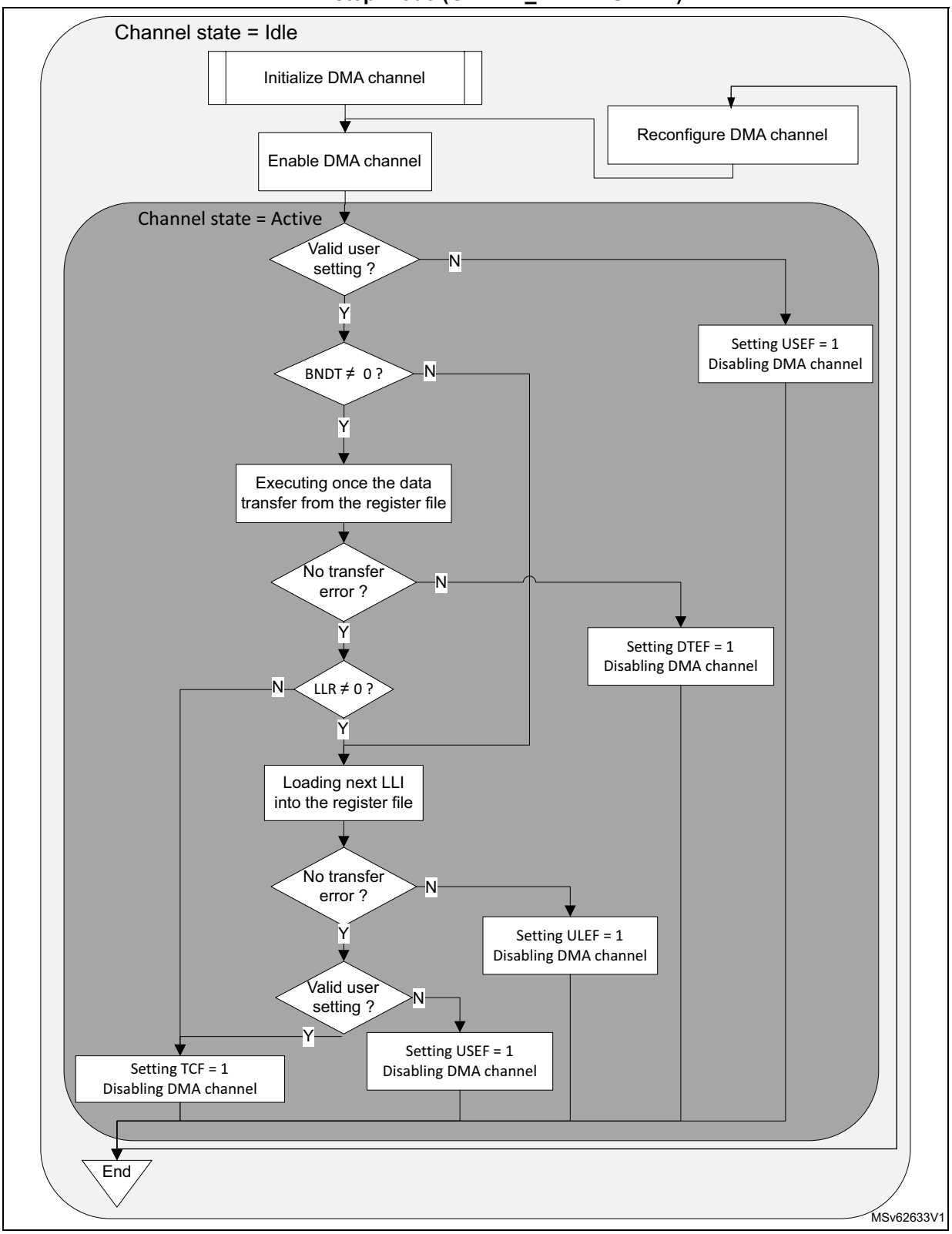
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 102 depicts the GPDMA channel execution mode, and its programming in link step mode.

Note: Figure 102 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 102, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).

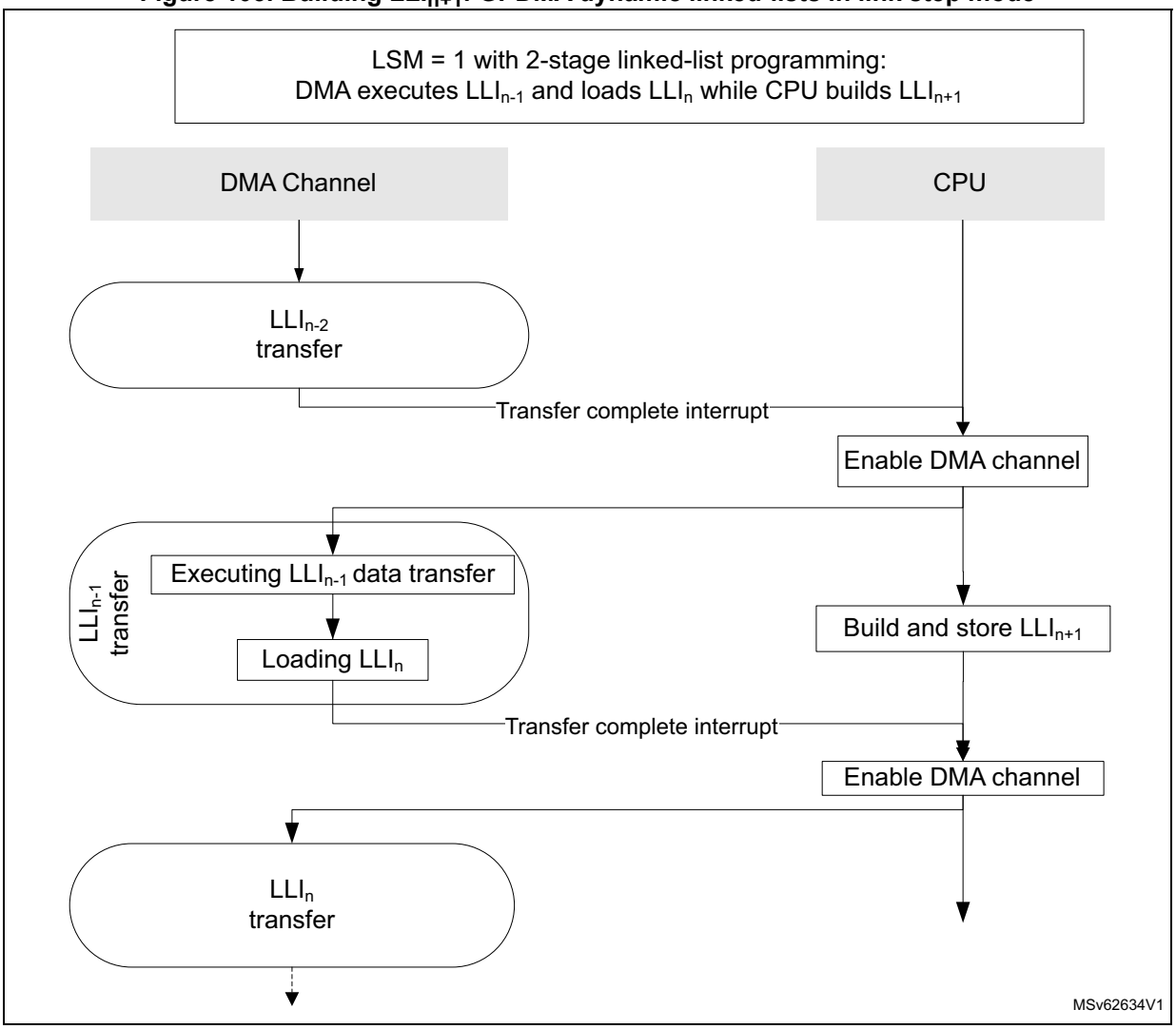
Figure 102. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in *Figure 103*.

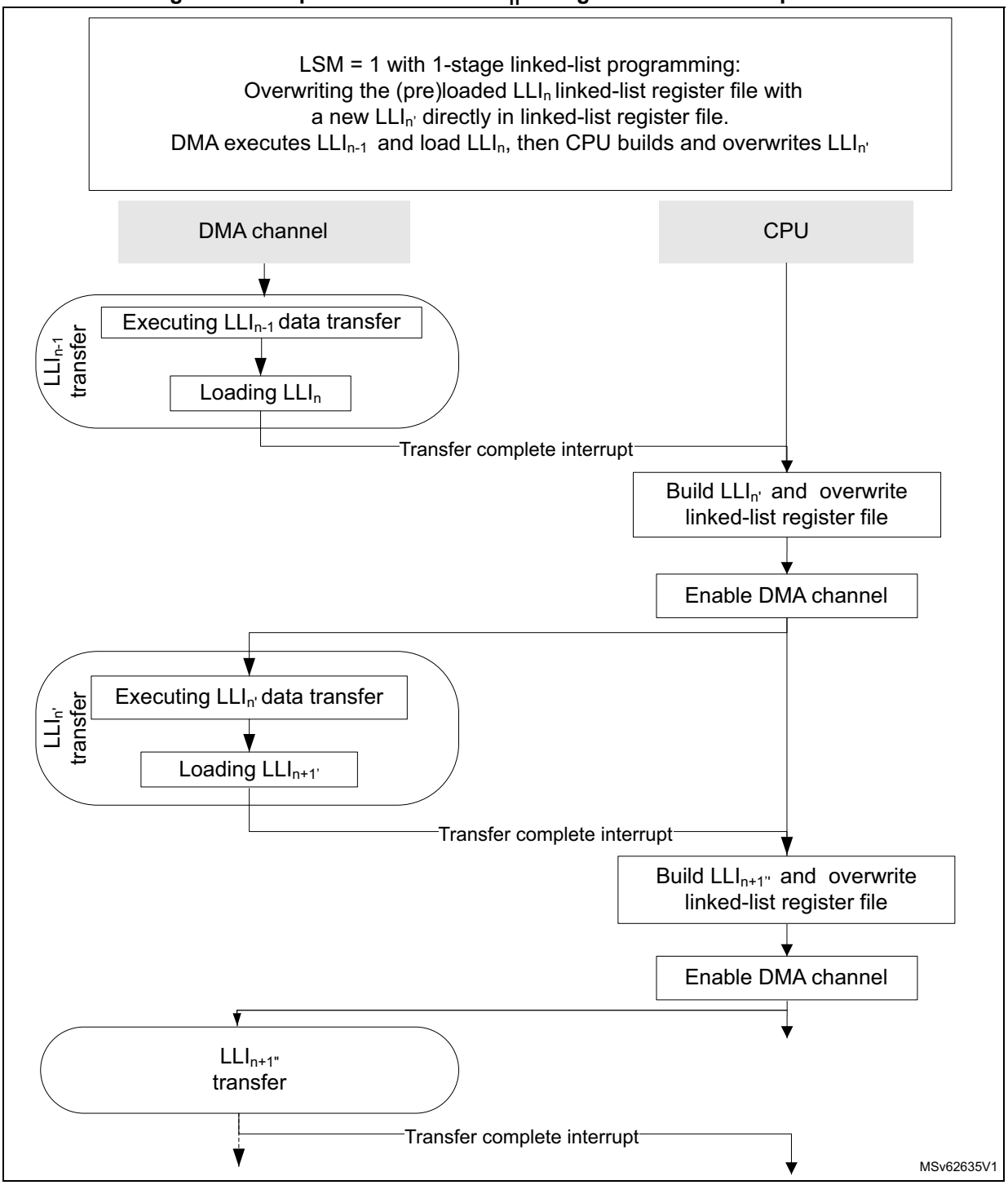
Figure 103. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in *Figure 104*.

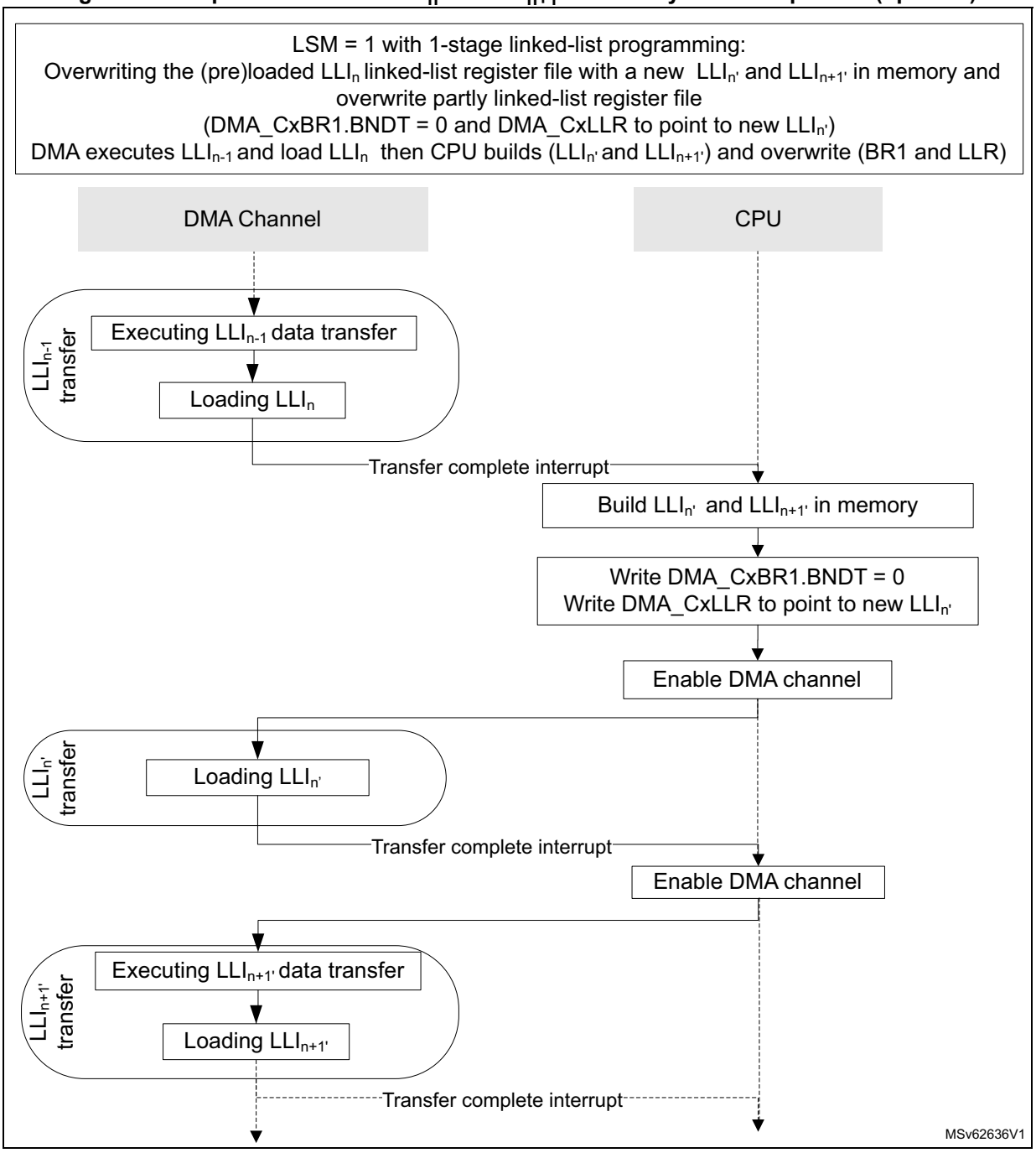
Figure 104. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n')}$ as shown in *Figure 105*.

Figure 105. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



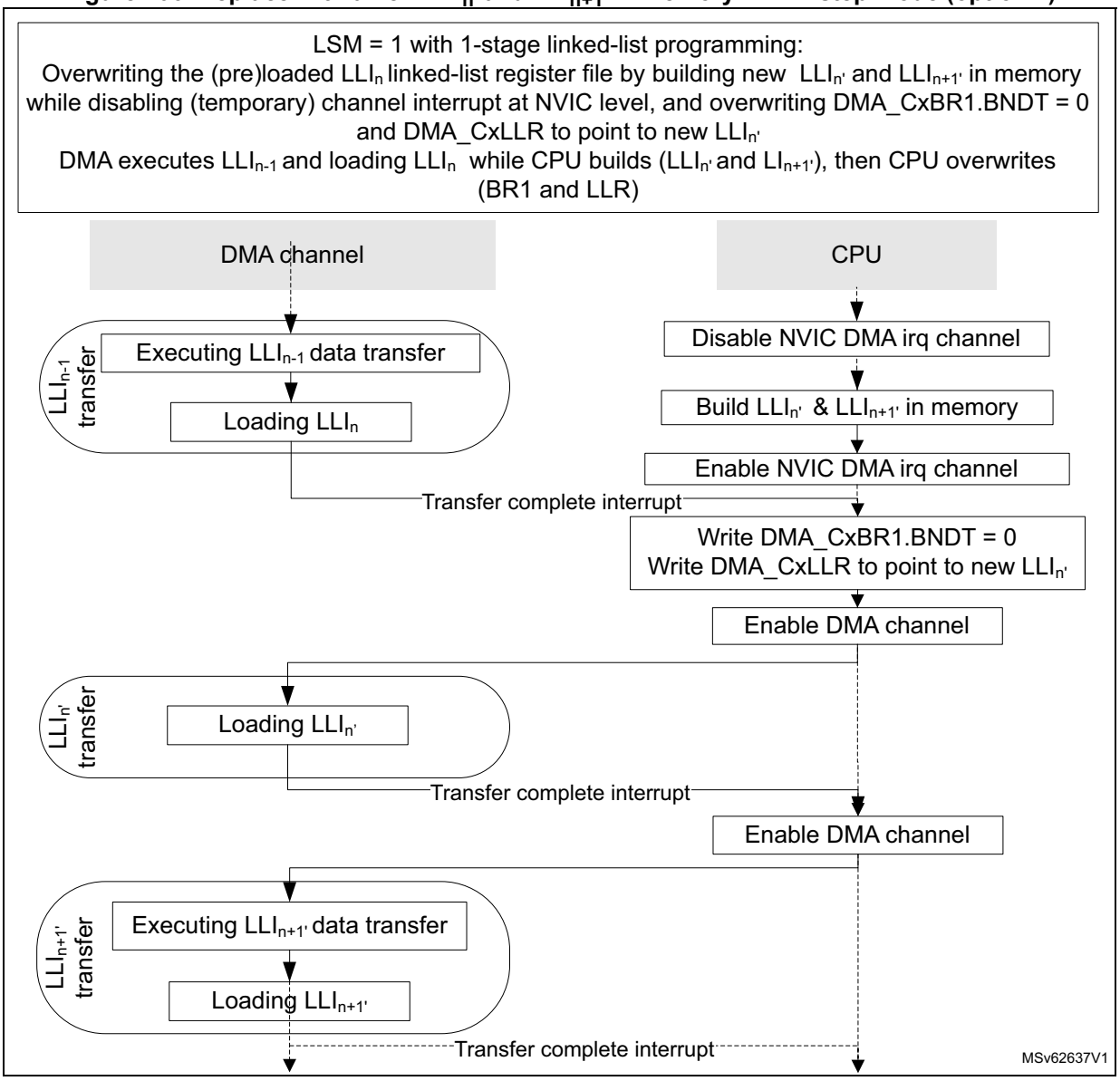
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile GPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 106*.

Figure 106. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



16.4.10 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the GPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 16.4.4*.)

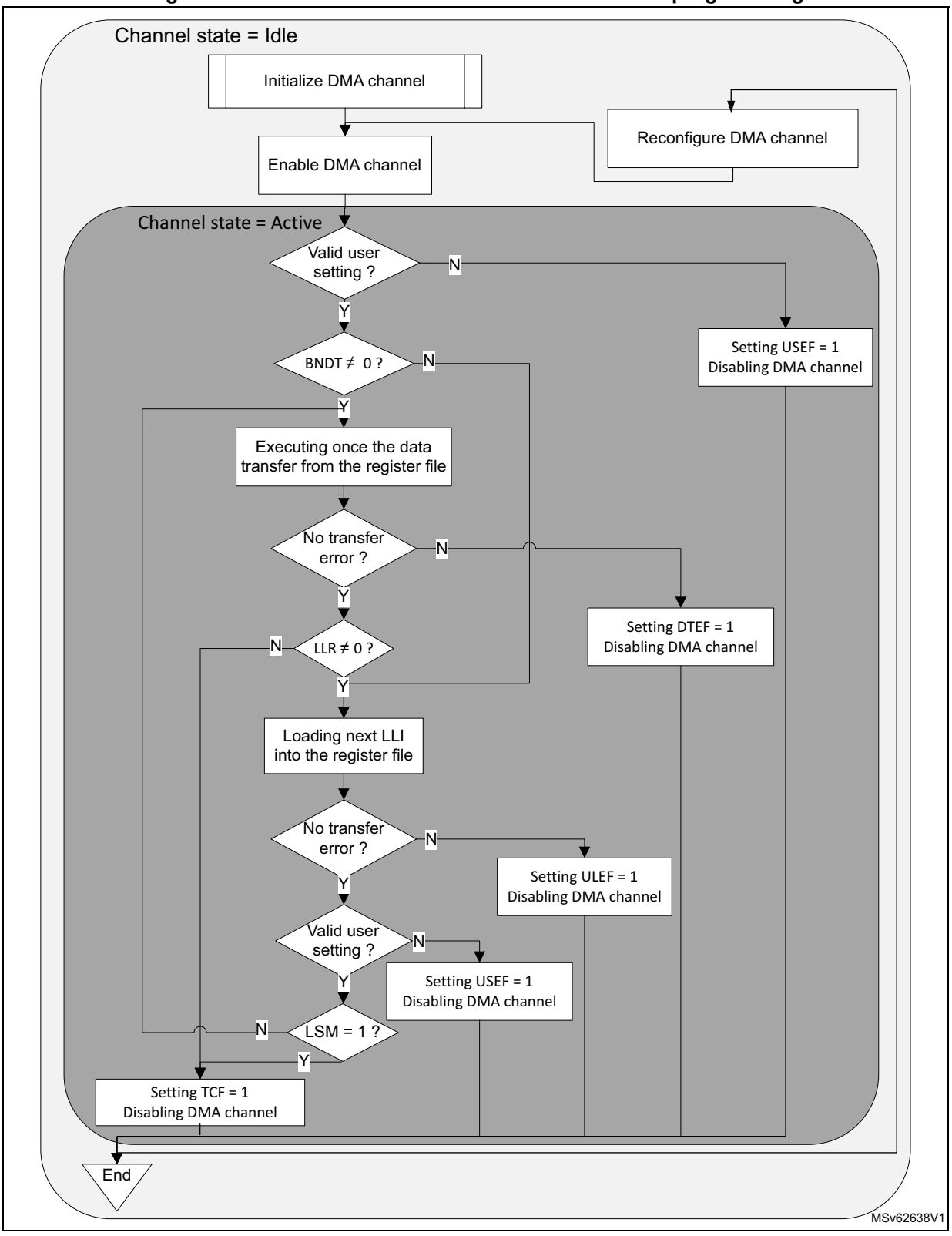
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 107 shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Note: Figure 107 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 107, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).

Figure 107. GPDMA channel execution and linked-list programming



16.4.11 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 16.3.2* for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

Table 156. Programmed GPDMA source/destination burst

SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/beats)	Next data/ beat address	Next burst address	Burst address alignment
00	1	0 (fixed)	n = 0 to 63(1)	n+1	+ 0	+ 0	1
01	2						2
10	4						4
00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
01	2				+ 2	+ 2 * (n + 1)	2
10	4				+ 4	+ 4 * (n + 1)	4
11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

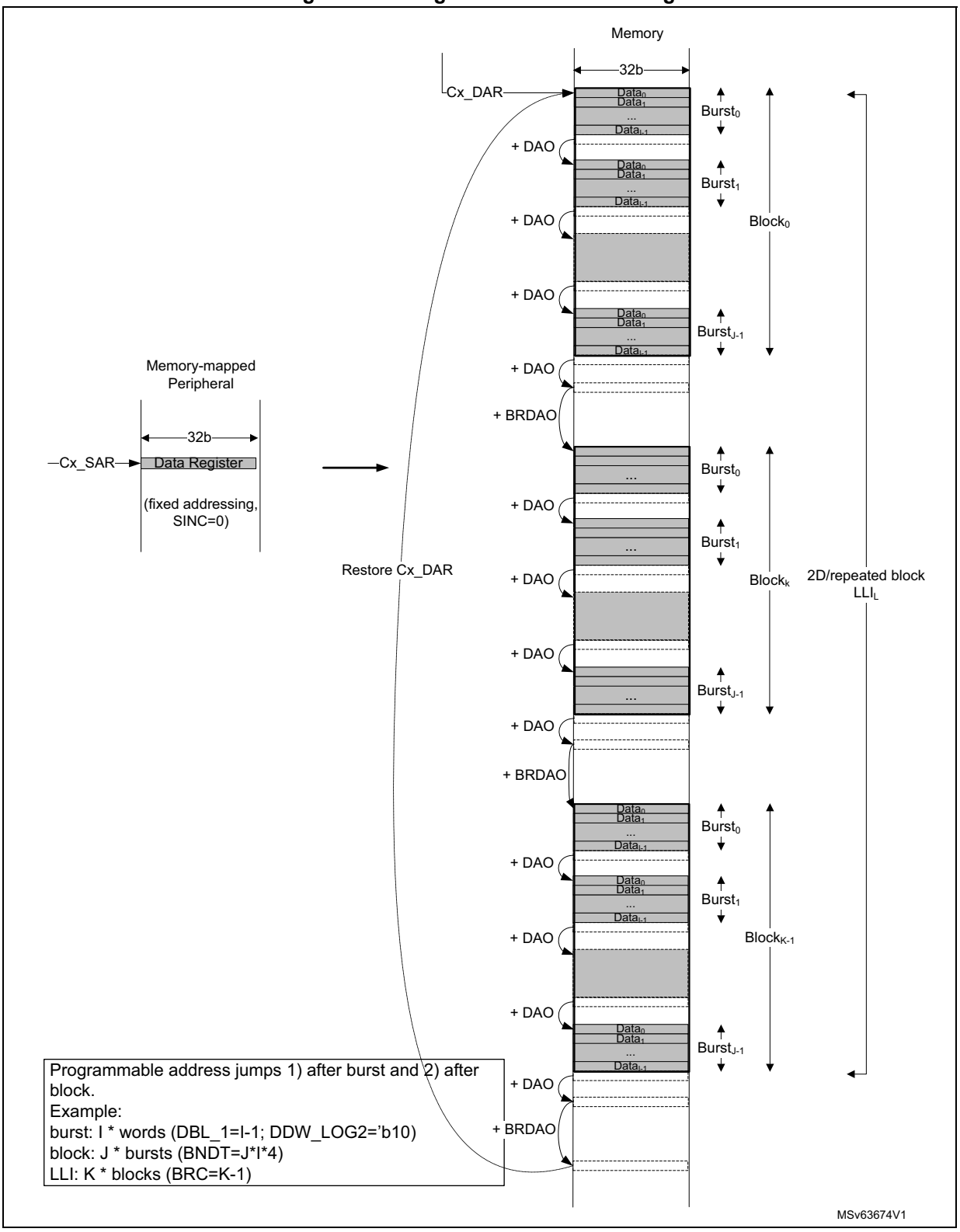
GPDMA burst with 2D addressing (channel x = 6 to 7)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 108* with a 2D destination buffer.

Figure 108. Programmed 2D addressing



GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is the half the FIFO size of channel x, in bytes (see *Section 16.3.2* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size. Each transfer is of a size that is lower than or equal to half of the FIFO size, without any user constraint.
- If the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields (see *Table 157*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 157. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)	0		$0B_3,0B_2,0B_1,0B_0$
							1		B_30,B_20,B_10,B_00
						01 (RA, SE)	0		SB_3,SB_2,SB_1,SB_0
							1		B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$000B_1,000B_0$
							1		$00B_10,00B_00$
							0	1	$0B_100,0B_000$
							1		B_1000,B_0000
						01 (RA, SE)	0	0	$SSSB_1,SSSB_0$
							1		SSB_1S,SSB_0S
							0	1	SB_1SS,SB_0SS
							1		B_1SSS,B_0SSS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$		00	Byte	00 (RA, LT)	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

Table 157. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	x	01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$00B_3B_2,00B_1B_0$
							1		$00B_2B_3,00B_0B_1$
							0	1	B_3B_200,B_1B_000
							1		B_2B_300,B_0B_100
						01 (RA, SE)	0	0	SSB_3B_2,SSB_1B_0
							1		SSB_2B_3,SSB_0B_1
							0	1	B_3B_2SS,B_1B_0SS
							1		B_2B_3SS,B_0B_1SS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						10 (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)	0		B_5B_4,B_1B_0
							1		B_4B_5,B_0B_1
						01 (LA, RT)	0		B_7B_6,B_3B_2
							1		B_6B_7,B_2B_3
						1x (UNPACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$

Table 157. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	10	Word	xx	0	0	$B_7B_6B_5B_4$,$B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5$,$B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6$,$B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7$,$B_0B_1B_2B_3$
			1	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						1x (UNPACK)			B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0
				01	Half-word	00 (RA, LT)	0		B_6B_4,B_2B_0
							1		B_4B_6,B_0B_2
						01 (LA, RT)	0		B_7B_5,B_3B_1
							1		B_5B_7,B_1B_3
						1x (UNPACK)	0		B_7B_5,B_6B_4,B_3B_1,B_2B_0
							1		B_5B_7,B_4B_6,B_1B_3,B_0B_2
				10	Word	xx	0	0	$B_7B_5B_6B_4$,$B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6$,$B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5$,$B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7$,$B_0B_2B_1B_3$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

16.4.12 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[7:0] field (see *Section 16.3.4*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 109*, is defined by:

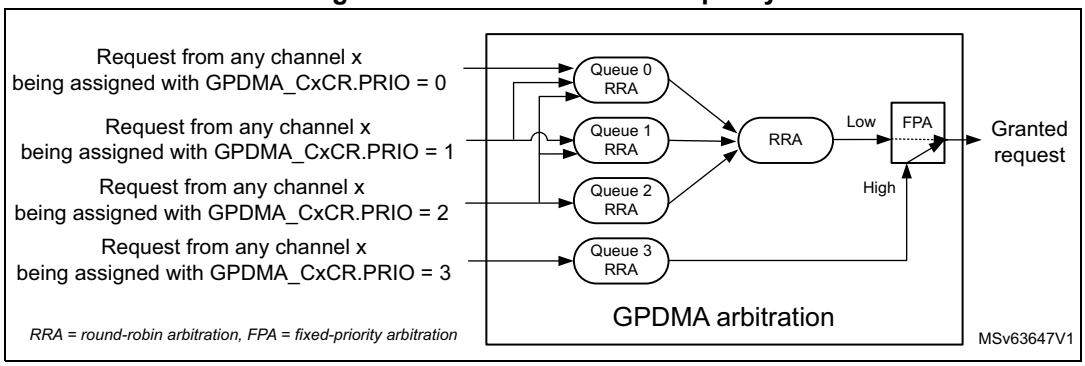
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1, or 2) for non time-sensitive channels with priority 0, 1, or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.

– Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
– Any queue 0, 1, or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
– Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1, and queue 2.

Figure 109. GPDMA arbitration policy



GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  – If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).
  $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).
  $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$
  $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).
  $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$
  $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$
  The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.

- **Example 2**: application where the user distributes a same non-null N number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is
    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is
    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is
    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is
    $B_{n, P0} = 1/N *1/9 * (1 - B_{Q3})$
    $B_{n, P1} = 1/N *5/18 * (1 - B_{Q3})$
    $B_{n, P2} = 1/N *11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for $J > 0$ (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ ($K>0$ in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 ($K > 0$)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 ($K = 0$)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

16.4.13 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see *Section 16.3.7* for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel x (x = 6 to 7), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in GPDMA_CxTR2 for more details).

Trigger hit memorization and trigger overrun flag generation

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

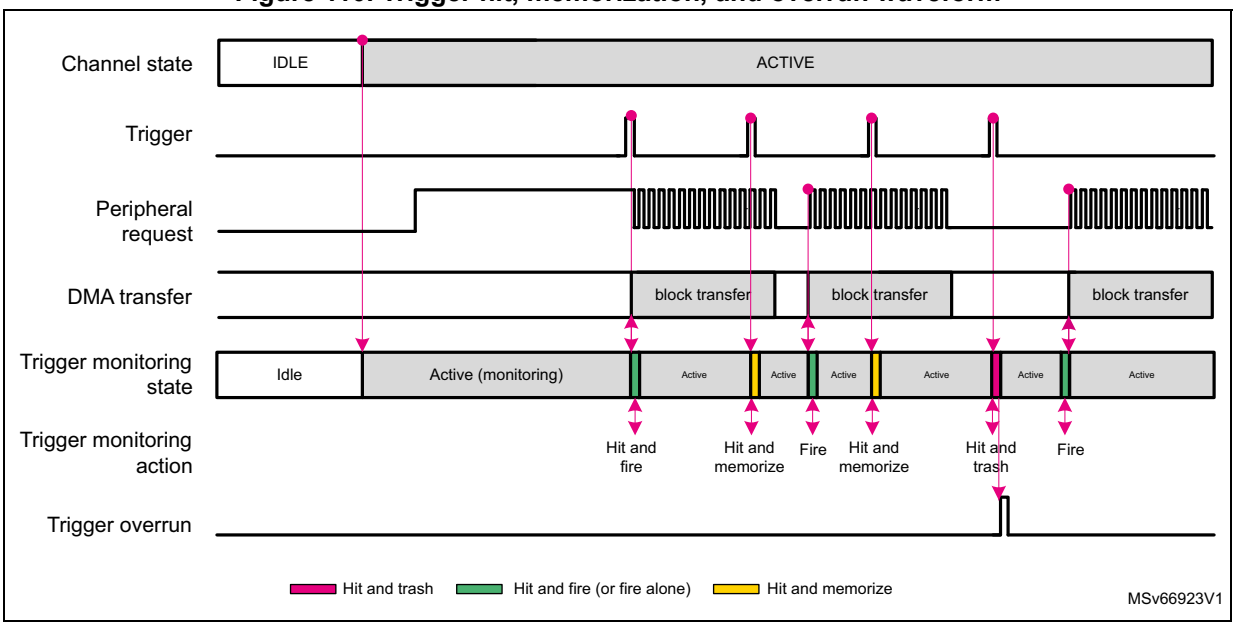
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Figure 110 illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 110. Trigger hit, memorization, and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

16.4.14 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

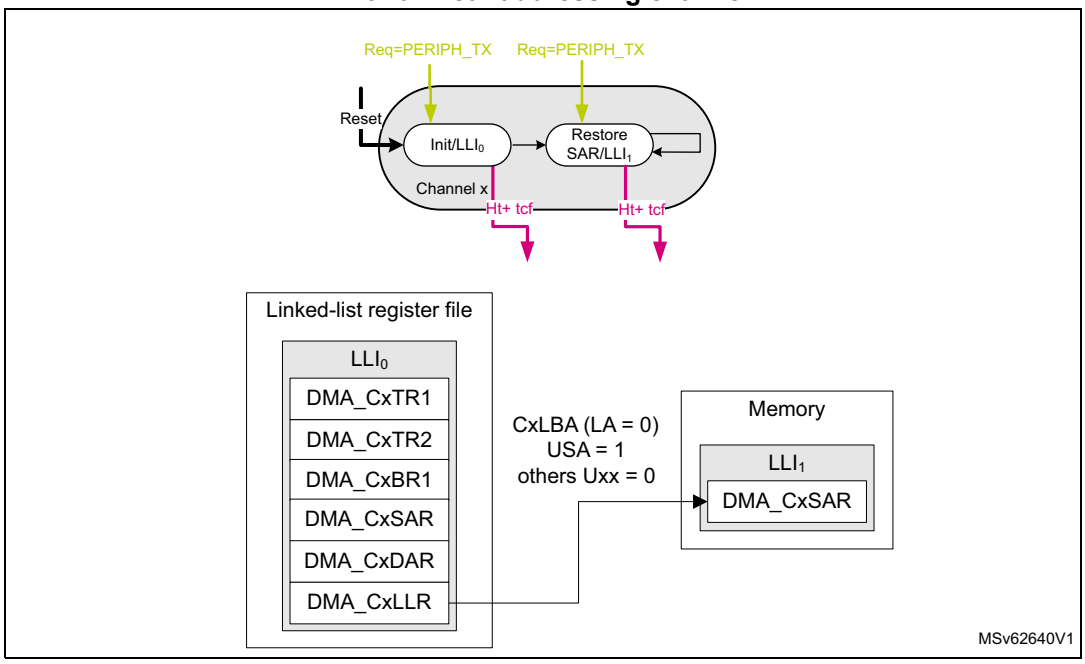
For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete transfer

events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

Figure 111 illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

Figure 111. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel

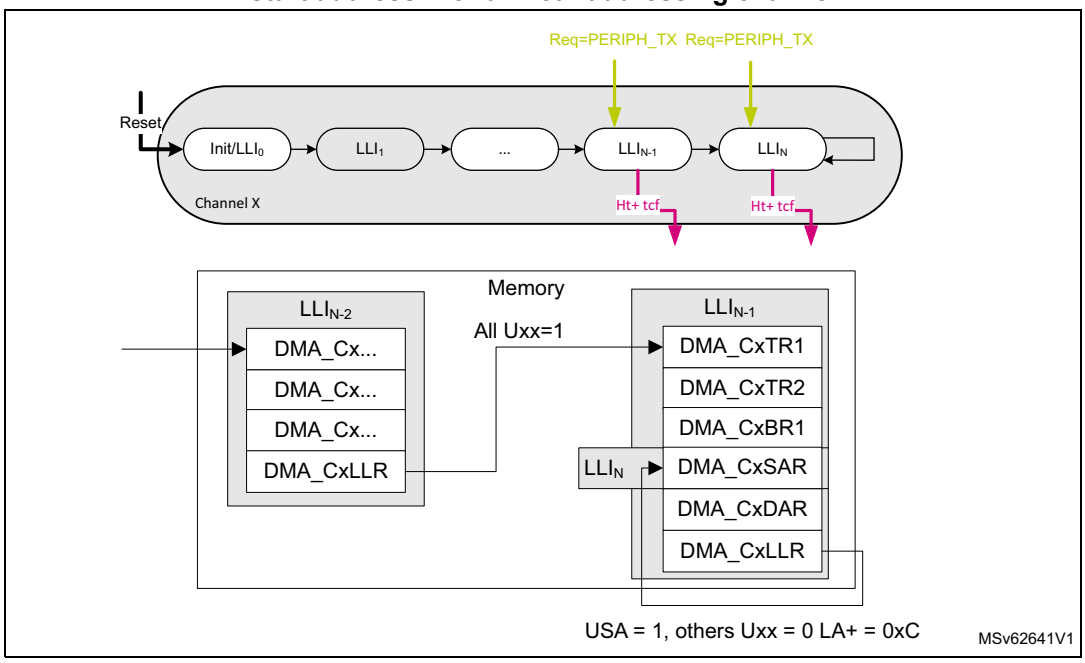


Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

Figure 112 illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 112. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



16.4.15 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 16.3.6*).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with a hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

16.4.16 GPDMA byte-based padding at the block level

For a given channel x, a LLI_n data transfer consists of a block transfer and is defined by the source block size GPDMA_CxBR1.BNDT[15:0], the size of the block of data to be transfered from the source, in bytes.

During a peripheral-to-memory transfer, a peripheral may be streaming and receiving a given number of bytes (for example, via an I2C, UART/USART/LPUART, ADC, DAC) or half-words, and this block size may not be a multiple of a word or a double-word. It is important to allow the user to program any number of bytes for the GPDMA source block size independently from the source data width and the destination data width, in order to use efficiently the bandwidth of the 32-bit AHB system bus and/or the 64-bit AXI system bus when accessing and writing into memory (embedded SRAM, NVM or external memory). This enables to decrease the bus bandwidth for the memory transfers by a ratio of up to 4 in AHB and up to 8 in AXI. More generally, having the source block size independent from the source data width and the destination data width can be useful also with memory-to-peripheral transfers, and with memory-to-memory transfers (AHB from/to AXI).

A FIFO-based channel is able to perform byte-based padding on the last write(s) up to the fully transferred source block size. Then channel FIFO is empty. Data handling is performed on zero bytes in the same way as on regular data bytes from the source, as explained in *GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment*.

There is byte-based padding at the block level:

1. If source data width is less than the destination data width, and if FIFO queuing and data packing is configured (PAM[1:0] = 1x):

- Padding with zero bytes. For example, if BNDT= 5, 8-bit reads, 32-bit writes:
  - if DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 000B_4, $B_3B_2B_1B_0$
  - if DBX = 0, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> 0$B_4$00, $B_1B_0B_3B_2$
  - if DBX = 1, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 00$B_4$0, $B_2B_3B_0B_1$
  - if DBX = 1, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> $B_4$000, $B_0B_1B_2B_3$

2. If source data width > destination data width, and if FIFO queuing and data unpacking is configured (PAM[1:0] = 1x):

- Padding with zero bytes.
- On the last read, only the number of LSB(s) up to the block size are written to the FIFO. The last written data to the destination is padded with zero bytes up to the destination data width. For example, if BNDT= 5, 32-bit reads, 16-bit writes:
  - if DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 0B_4, B_3B_2, B_1B_0
  - if DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$, -> $B_4$0, B_2B_3, B_0B_1

3. If source data width = destination data width (and BNDT[15:0] is not a multiple of that value):

- Padding with zero bytes. For example, if BNDT= 6, 32-bit reads, 32-bit writes are configured
  - if SBX = 0, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_5B_4$, $B_3B_2B_1B_0$
  - if SBX = 0, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_5B_400, $B_1B_0B_3B_2$
  - if SBX = 0, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_4B_5$, $B_2B_3B_0B_1$
  - if SBX = 0, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4B_500, $B_0B_1B_2B_3$
  - if SBX = 1, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $000B_4$, $B_3B_1B_2B_0$
  - if SBX = 1, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_400$, $B_2B_0B_3B_1$
  - if SBX = 1, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_40$, $B_1B_3B_0B_2$
  - if SBX = 1, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4000, $B_0B_2B_1B_3$

4. If source data width < destination data width, and if destination data is configured as right-aligned and, either configured with zero padding bytes (PAM[1:0] = 00), or configured with sign extension (PAM[1:0] = 01):

- Padding with zero bytes if PAM[1:0] = 00. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 00:
  - if DBX =0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $000B_4$, $000B_3$, $000B_2$, $000B_1$, $000B_0$
- Padding with sign extension if PAM[1:0] = 01. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 01:
  - if DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $SSSB_4$, $SSSB_3$, $SSSB_2$, $SSSB_1$, $SSSB_0$

5. If source data width > destination data width, and if destination data is configured as right-aligned (PAM[1:0] = 00), or left-aligned (PAM[1:0] = 01):

- Padding with zero bytes. For example, BNDT = 5, 32-bit reads, 16-bit writes:
  - if SBX = 0, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_1B_0
  - if SBX = 0, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_1
  - if SBX = 0, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_2
  - if SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_2B_3
  - if SBX = 1, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_2B_0
  - If SBX = 1, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_2
  - If SBX = 1, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_1
  - If SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_1B_3

16.4.17 GPDMA secure/nonsecure channel

The GPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel x is a secure or a nonsecure hardware resource, as configured by GPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A nonsecure read access to a register field of this channel is forced to return 0, except for GPDMA_SECCFGR, GPDMA_PRIVCFGR, GPDMA_RCFGLOCKR that are readable by a nonsecure agent.
- A nonsecure write access to a register field of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or nonsecure the GPDMA data transfer from the source (GPDMA_CxTR1.SSEC) and the GPDMA data transfer to the destination (GPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with secure transfers via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a secure bus that reflects GPDMA_SECCFGR, to keep the other peripherals informed of the secure/nonsecure state of each GPDMA channel x.

The GPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure GPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transfer as a nonsecure software is not allowed to do so and must have GPDMA_CxCR.EN = 0 before the nonsecure software can reprogram the GPDMA_CxCR for a next transfer. The secure software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

16.4.18 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for GPDMA_PRIVCFGR, GPDMA_SECCFGR, GPDMA_RCFGLOCKR that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the

privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

16.4.19 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single or a burst) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a GPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).

- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1. BNDT = 0)

16.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

16.6 GPDMA in low-power modes

Table 158. Effect of low-power modes on GPDMA

Mode	Description
Sleep	No effect. GPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the GPDMA registers is kept when entering Stop mode.
Standby	The GPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 16.3.3* to know if any Stop mode is supported.

16.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 159. GPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
GPDMA_CHx	Transfer complete	GPDMA_CxCR.TCIE	GPDMA_CxSR.TCF	Write 1 to GPDMA_CxFCR.TCF
	Half transfer	GPDMA_CxCR.HTIE	GPDMA_CxSR.HTF	Write 1 to GPDMA_CxFCR.HTF
	Data transfer error	GPDMA_CxCR.DTEIE	GPDMA_CxSR.DTEF	Write 1 to GPDMA_CxFCR.DTEF
	Update link error	GPDMA_CxCR.ULEIE	GPDMA_CxSR.ULEF	Write 1 to GPDMA_CxFCR.ULEF
	User setting error	GPDMA_CxCR.USEIE	GPDMA_CxSR.USEF	Write 1 to GPDMA_CxFCR.USEF
	Suspended	GPDMA_CxCR.SUSPIE	GPDMA_CxSR.SUSPF	Write 1 to GPDMA_CxFCR.SUSPF
	Trigger overrun	GPDMA_CxCR.TOFIE	GPDMA_CxSR.TOF	Write 1 to GPDMA_CxFCR.TOF

A GPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

- A half-transfer event is a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see *Section 16.4.19*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

16.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

16.8.1 GPDMA secure configuration register (GPDMA_SECCFGR)

Address offset: 0x00

Reset value: 0x0000 0000

A write access to this register must be secure and privileged. A read access is secure or nonsecure, privileged or unprivileged.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (when GPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SECx**: secure state of channel x (x = 7 to 0)
0: Nonsecure
1: Secure

16.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure.

This register can mix secure and nonsecure information. If a channel x is configured as secure (GPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN =1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRIVx**: privileged state of channel x (x = 7 to 0)
0: unprivileged
1: privileged

16.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attributes in order to individually lock, for example at boot time, the secure privileged attributes of any GPDMA

channel/resource (to lock the setting of GPDMA_CxSECCFGR, GPDMA_CxPRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure.

Note: If TZEN = 0, this register cannot be written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
								rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LOCKx**: lock the configuration of GPDMA_SECCFGR.SECx, GPDMA_PRIVCFGR.PRIVx until a global GPDMA reset (x = 7 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

16.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with GPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of GPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of GPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in GPDMA_CxFCR.

If a channel x is in secure state (GPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **MISx**: masked interrupt status of channel x (x = 7 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

16.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR)

Address offset: 0x10

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (GPDMA_SECCFGR.SECx = 1). It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **MISx**: masked interrupt status of the secure channel x (x = 7 to 0)
0: no interrupt occurred on the secure channel x
1: an interrupt occurred on the secure channel x

16.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (GPDMA_SECCFGR.SECx).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LBA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **LBA[31:16]**: linked-list base address of GPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

16.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C+ 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: corresponding TOF flag cleared

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half-transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.

16.8.8 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAINING F	IDLEF
	r	r	r	r	r	r	r							r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level (in bytes)
Number of available bytes in the FIFO.
Note: After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag
0: no trigger overrun event
1: a trigger overrun event occurred

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half-transfer flag
0: no half-transfer event
1: a half-transfer event occurred
A half-transfer event is either a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).
A half-block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.
A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2)) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DRAININGF**: FIFO draining flag
0: No received order to drain the FIFO or channel FIFO draining completed
1: Ongoing channel FIFO draining and channel termination
This flag is asserted after that the order to drain the FIFO is received, and meanwhile that the channel is enabled and draining the FIFO and before the channel is terminated with a drained FIFO.
This flag is deasserted by hardware when the enabled channel has been ordered to be terminated with FIFO draining, and the hardware effectively terminates.

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is deasserted by hardware when the channel is enabled (GPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

16.8.9 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM
								rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAIN FIFO	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw					w	rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** priority level of the channel x GPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: linked-list allocated port
This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed **once** for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half-transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DRAINFIFO:** empty FIFO
This bit is write only. Writing 0 has no impact.
Writing 1 to RESET in this register causes the hardware to deassert this DRAINFIFO bit, whatever is written into this DRAINFIFO. Else:
Software must write 1 in order to drain the channel FIFO on a suspended channel.
If the software concurrently writes 1 to DRAINFIFO and 1 to SUSP and 0 to reset, the hardware suspends the channel and drains the FIFO by popping out the residual FIFO bytes to the destination (see the programming sequence in *Figure 95*).
0: Write: no action
1: Write: drain FIFO after having suspended the channel

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 93*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset

This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP, DRAINFIFO and EN bits (whatever is written receptively in bit 3 and bit 2 and bit 0).

The reset is effective when the channel is in steady state, meaning one of the following:

- active channel in suspended state (GPDMA_CxSR.SUSPF = 1, GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1 and GPDMA_CxSR.DRAININGF = 0)
- channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).

After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 94*).

0: no channel reset

1: channel reset

Bit 0 **EN:** enable

Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:

this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: write: ignored, read: channel disabled

1: write: enable channel, read: channel enabled

16.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or unprivileged, depending on the privileged state of the channel x in GPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **DSEC**: security attribute of the GPDMA transfer to the destination

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer to the destination is nonsecure.

Bit 30 **DAP**: destination allocated port

This bit is used to allocate the master port for the destination transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **DHX**: destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word:

0: no halfword-based exchanged within word

1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0:no byte-based exchange within half-word

1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: destination incrementing burst

0: fixed burst

1: contiguously incremented burst

The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and if present address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

Bit 15 **SSEC**: security attribute of the GPDMA transfer from the source
If GPDMA_SECCFGR.SECx = 1 and the access is secure:
0: GPDMA transfer nonsecure
1: GPDMA transfer secure
This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.
When GPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer from the source is nonsecure.

Bit 14 **SAP**: source allocated port
This bit is used to allocate the master port for the source transfer
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: source byte exchange within the unaligned half-word of each source word
If the source data width is shorter than a word, this bit is ignored.
If the source data width is a word:
0:no byte-based exchange within the unaligned half-word of each source word
1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: padding/alignment mode

If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.

Condition: PAM_1

00: source data is transferred as right aligned, padded with 0s up to the destination data width

01: source data is transferred as right aligned, sign extended up to the destination data width

10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer

Condition: PAM_2

00: source data is transferred as right aligned, left-truncated down to the destination data width

01: source data is transferred as left-aligned, right-truncated down to the destination data width

10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: source incrementing burst

0: fixed burst

1: contiguously incremented burst

The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a burst in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

16.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x. This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]					
rw	rw					rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half-transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

01: channel x (x = 0 to 5), same as 00, channel x (x = 6 to 7), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half-transfer event is generated at the half of the LLI data transfer The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x = 6 to 7), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half-transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half-transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 16.3.7*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x (x = 6 to 7): same, each block also if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x (x = 0 to 5): same as 00; channel x (x = 6 to 7): at 2D/repeated block level: the first burst read of the 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. LLI data transfer is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger:

– If the peripheral is the source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is the destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and one trigger hit.

As shown in *Figure 110: Trigger hit, memorization, and overrun waveform*, the GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10). The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled. After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 16.3.2 for the list of the implemented channels with this feature).

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see *Section 16.3.6* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 16.3.4*).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[7:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: GPDMA hardware request selection

These bits are ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1(software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 16.3.4*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active GPDMA channels (GPDMA_CxCR.EN = 1 and GPDMA_CxTR2.SWREQ = 0 for these channels). GPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

16.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

16.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]										
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.
– If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
– if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
– if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

16.8.14 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 6 to 7). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

16.8.15 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].

– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 6 to 7), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

16.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DAO[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SAO[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: destination address offset increment

The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: source address offset increment

The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

16.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRSAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

16.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory

This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.

0: no GPDMA_CxDAR update

1: GPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory

This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.

0: no GPDMA_CxLLR update

1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure

If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.

Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register GPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

16.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update

1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update

1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update

1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0:no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register GPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

16.8.20 GPDMA register map

Table 160. GPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPDMA_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																									0	0	0	0	0	0	0	0
0x04	GPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																									0	0	0	0	0	0	0	0
0x08	GPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																									0	0	0	0	0	0	0	0

Table 160. GPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	GPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									0	0	0	0	0	0	0	0
0x10	GPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									0	0	0	0	0	0	0	0
0x14 - 0x4C	Reserved	Reserved																															
0x50+0x80 * x (x=0 to 7)	GPDMA_ CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x54 to 0x58+0x80*x (x=0 to 7)	Reserved	Reserved																															
0x5C+0x80 * x (x=0 to 7)	GPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x60+ 0x80 * x (x=0 to 7)	GPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]								Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAININGF	IDLEF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0							0	1
0x64+ 0x80 * x (x=0 to 7)	GPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAINFIFO	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0					0	0	0	0
0x68 to 0x8C+0x80 * x (x=0 to 7)	Reserved	Reserved																															
0x90+0x80 * x (x=0 to 7)	GPDMA_CxTR1	DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2 [1:0]		SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2 [1:0]	
	Reset value	0	0			0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0
0x94+0x80 * x (x=0 to 7)	GPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]						TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
	Reset value	0	0					0	0			0	0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0
0x98+0x80 * x (x=0 to 5)	GPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98+0x80 * x (x=6 to 7)	GPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C+0x80 * x (x=0 to 7)	GPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0+0x80 * x (x=0 to 7)	GPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4+0x80 * x (x=6 to 7)	GPDMA_CxTR3	Res.	Res.	Res.	DAO[12:0]													Res.	Res.	Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8+0x80 * x (x=6 to 7)	GPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 160. GPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xCC+0x80 * x (x=0 to 5)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xCC+0x80 * x (x=6 to 7)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

17 General purpose direct memory access controller (GPDMA)

This section applies only to STM32G53x/54x devices.

17.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

17.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half-transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 6 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 17.3.2*)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level and byte-based padding

  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers
  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 17.3.2*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Programmable GPDMA request and trigger selection
  - Programmable GPDMA half-transfer and transfer complete events generation
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers
  - Channel abort and restart
  - Channel terminate with FIFO draining
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and nonsecure GPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any GPDMA secure resource (register, register field) from a nonsecure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

17.3 GPDMA implementation

17.3.1 GPDMA instances

There are two instances of the GPDMA in the devices, named as GPDMA1 and GPDMA2.

Each GPDMA instance has the same channel-based implementation and is connected to the same requests and triggers, as detailed in the following sub-sections.

17.3.2 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

Table 161. GPDMA1/2 channel implementation

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 3	2	0	Channel x (x = 0 to 3) is implemented with: – a FIFO of 8 bytes, 2 words – fixed/contiguously incremented addressing These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and SRAM.
x = 4 to 5	4	1	Channel x (x = 4 to 5) is implemented with: – a FIFO of 32 bytes, 8 words – 2D addressing These channels may be also used for GPDMA transfers between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.

17.3.3 GPDMA in low-power modes

The GPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 162. GPDMA1/2 wake-up in low-power modes

Feature	Low-power modes
Wake-up	GPDMA1/2 in Sleep mode

17.3.4 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[7:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 163. Programmed GPDMA1/2 request

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
0	adc1_dma
1	adc2_dma
2	dac1_ch1_dma
3	dac2_ch1_dma
4	tim6_upd_dma
5	tim7_upd_dma
6	spi1_rx_dma
7	spi1_tx_dma
8	spi2_rx_dma

Table 163. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
9	spi2_tx_dma
10	Reserved
11	Reserved
12	i2c1_rx_dma
13	i2c1_tx_dma
14	Reserved
15	i2c2_rx_dma
16	i2c2_tx_dma
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	usart1_rx_dma
22	usart1_tx_dma
23	usart2_rx_dma
24	usart2_tx_dma
25	Reserved
26	Reserved
27	uart4_rx_dma
28	uart4_tx_dma
29	uart5_rx_dma
30	uart5_tx_dma
31	uart6_rx_dma
32	uart6_tx_dma
33	Reserved
34	Reserved
35	Reserved
36	Reserved
37	hrtim1_master_dma
38	hrtim1_timA_dma
39	hrtim1_timB_dma
40	hrtim1_timC_dma
41	hrtim1_timD_dma
42	hrtim1_timE_dma
43	hrtim1_timF_dma

Table 163. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
44	Reserved
45	lpuart1_rx_dma
46	lpuart1_tx_dma
47	Reserved
48	Reserved
49	Reserved
50	Reserved
51	Reserved
52	Reserved
53	Reserved
54	Reserved
55	Reserved
56	Reserved
57	Reserved
58	tim1_cc1_dma
59	tim1_cc2_dma
60	tim1_cc3_dma
61	tim1_cc4_dma
62	tim1_upd_dma
63	tim1_trgi_dma
64	tim1_com_dma
65	tim8_cc1_dma
66	tim8_cc2_dma
67	tim8_cc3_dma
68	tim8_cc4_dma
69	tim8_upd_dma
70	tim8_trgi_dma
71	tim8_com_dma
72	tim2_cc1_dma
73	tim2_cc2_dma
74	tim2_cc3_dma
75	tim2_cc4_dma
76	tim2_upd_dma
77	tim2_trgi_dma
78	Reserved

Table 163. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
79	Reserved
80	Reserved
81	Reserved
82	Reserved
83	Reserved
84	Reserved
85	Reserved
86	Reserved
87	Reserved
88	Reserved
89	Reserved
90	Reserved
91	Reserved
92	Reserved
93	Reserved
94	Reserved
95	Reserved
96	tim15_cc1_dma
97	tim15_cc2_dma
98	tim15_upd_dma
99	tim15_trgi_dma
100	tim15_com_dma
101	tim16_cc1_dma
102	tim16_upd_dma
103	tim17_cc1_dma
104	tim17_upd_dma
105	lptim1_ic1_dma
106	lptim1_ic2_dma
107	lptim1_ue_dma
108	Reserved
109	Reserved
110	Reserved
111	aes_out_dma
112	aes_in_dma
113	hash1_in_dma

Table 163. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
114	Reserved
115	Reserved
116	cordic1_rd_dma
117	cordic1_wr_dma
118	fmac1_rd_dma
119	fmac1_wr_dma
120	Reserved
121	Reserved
122	i3c1_rx_dma
123	i3c1_tx_dma
124	i3c1_tc_dma
125	i3c1_rs_dma
126	i2c4_rx_dma
127	i2c4_tx_dma
128	fmac2_rd_dma
129	fmac2_wr_dma
130	Reserved
131	Reserved
132	Reserved
133	Reserved
134	Reserved
135	dac5_ch1_dma
136	dac6_ch1_dma
137	dac7_ch1_dma
138	dac8_ch1_dma
139	dac9_ch1_dma
140	dac10_ch1_dma
141	Reserved
142	Reserved
143	Reserved
144	Reserved
145	Reserved
146	Reserved
147	Reserved
148	Reserved

Table 163. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
149	Reserved
150	Reserved
151	Reserved
152	Reserved
153	Reserved
154	Reserved
155	tim26_upd_dma
156	tim27_upd_dma

17.3.5 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 164. Programmed GPDMA1/2 request as a block request

GPDMA block requests
lptim1_ue_dma

17.3.6 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 165. GPDMA1/2 channel with peripheral early termination

GPDMA channel x with peripheral early termination
x = 0

This GPDMA support is activated when the channel x is programmed with GPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before the GPDMA has transferred the whole block (see *Section 17.4.15* for more details).

Table 166. Programmed GPDMA1/2 request with peripheral early termination

Programmed GPDMA channel x request with peripheral early termination
i3c1_rx_dma

17.3.7 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[5:0] in GPDMA_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 167. Programmed GPDMA1/2 trigger

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6
7	exti7
8	tamp_trgi1
9	tamp_trgi2
10	Reserved
11	lptim1_ch1
12	lptim1_ch2
13	Reserved
14	Reserved
15	rtc_alra_trgi
16	rtc_alrb_trgi
17	rtc_wut_trgi
18	gpdma1_ch0_tc
19	gpdma1_ch1_tc
20	gpdma1_ch2_tc
21	gpdma1_ch3_tc
22	gpdma1_ch4_tc
23	gpdma1_ch5_tc
24	Reserved
25	Reserved
26	gpdma2_ch0_tc
27	gpdma2_ch1_tc
28	gpdma2_ch2_tc
29	gpdma2_ch3_tc
30	gpdma2_ch4_tc

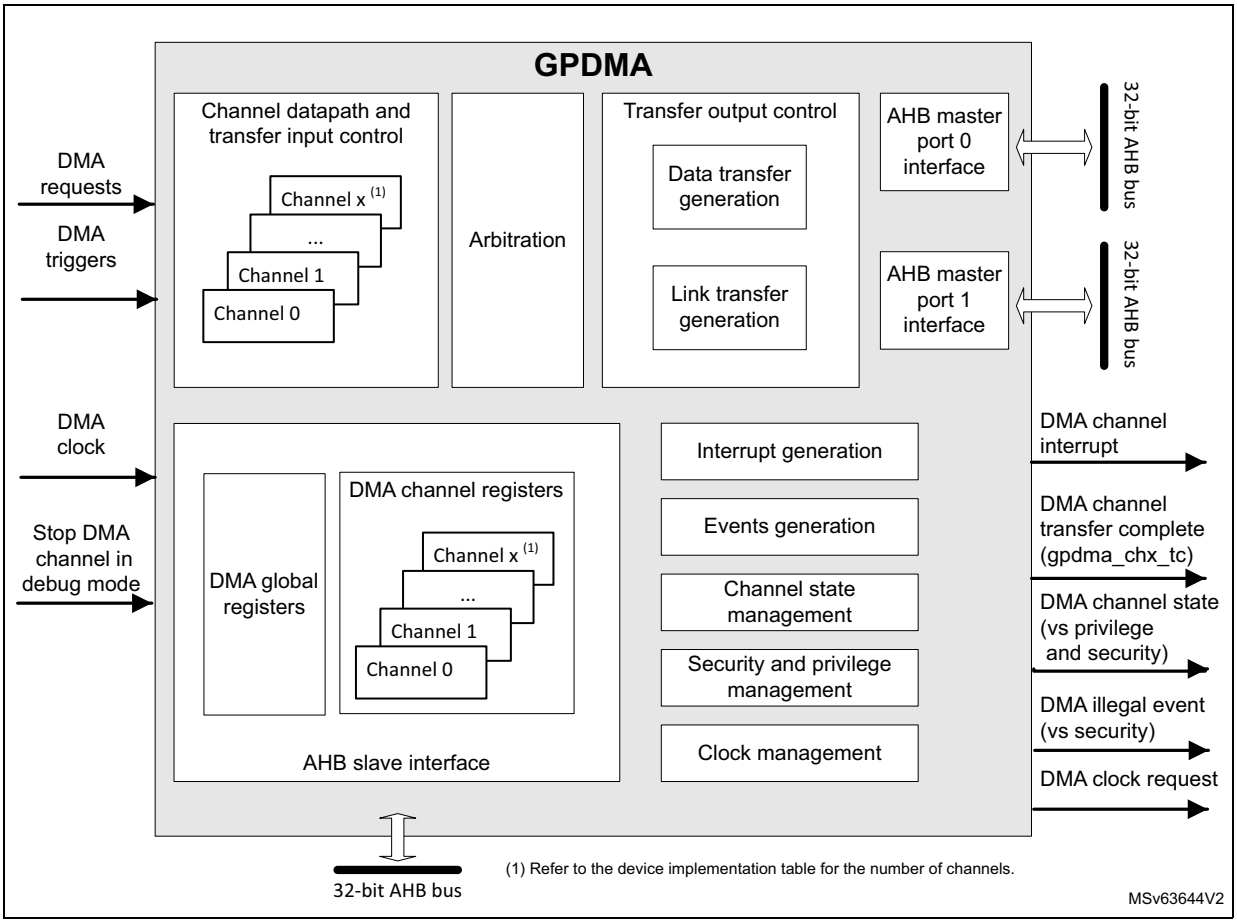
Table 167. Programmed GPDMA1/2 trigger (continued)

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
31	gpdma2_ch5_tc
32	Reserved
33	Reserved
34	tim2_tgo
35	tim15_tgo
36	tim12_tgo
37	Reserved
38	tim9_tgo
39	tim7_tgo
40	tim6_tgo

17.4 GPDMA functional description

17.4.1 GPDMA block diagram

Figure 113. GPDMA block diagram



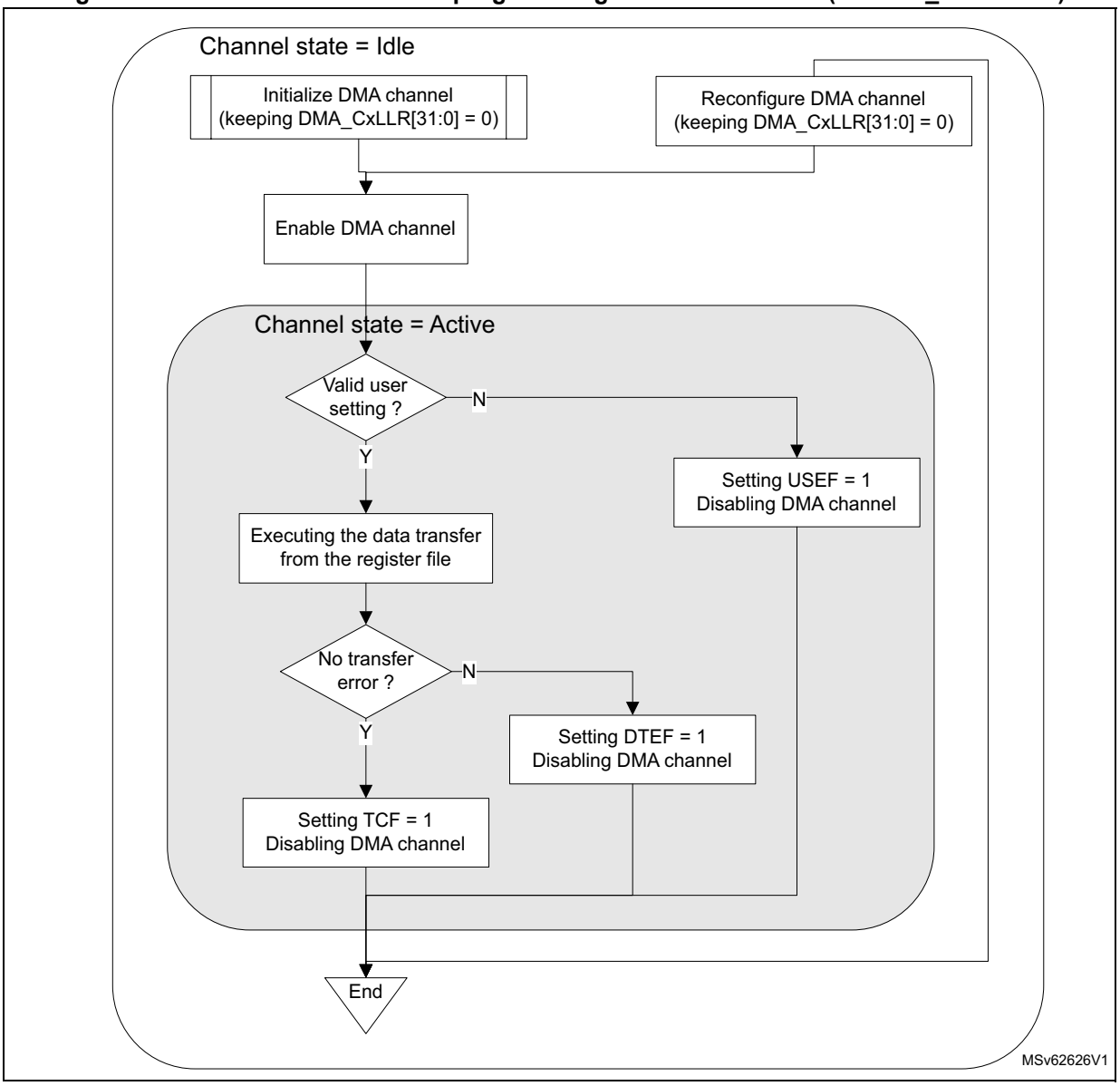
17.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

Figure 114 illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 114. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



17.4.3 GPDMA channel suspend and resume

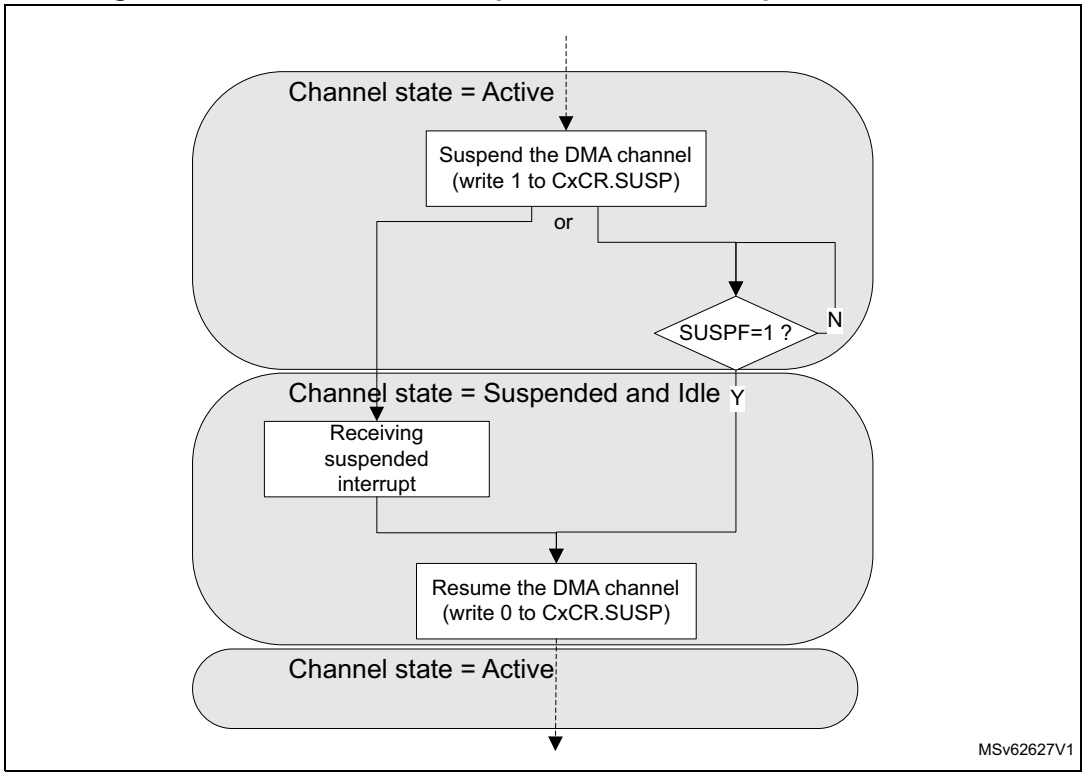
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note: An ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

Figure 115. GPDMA channel suspend and resume sequence



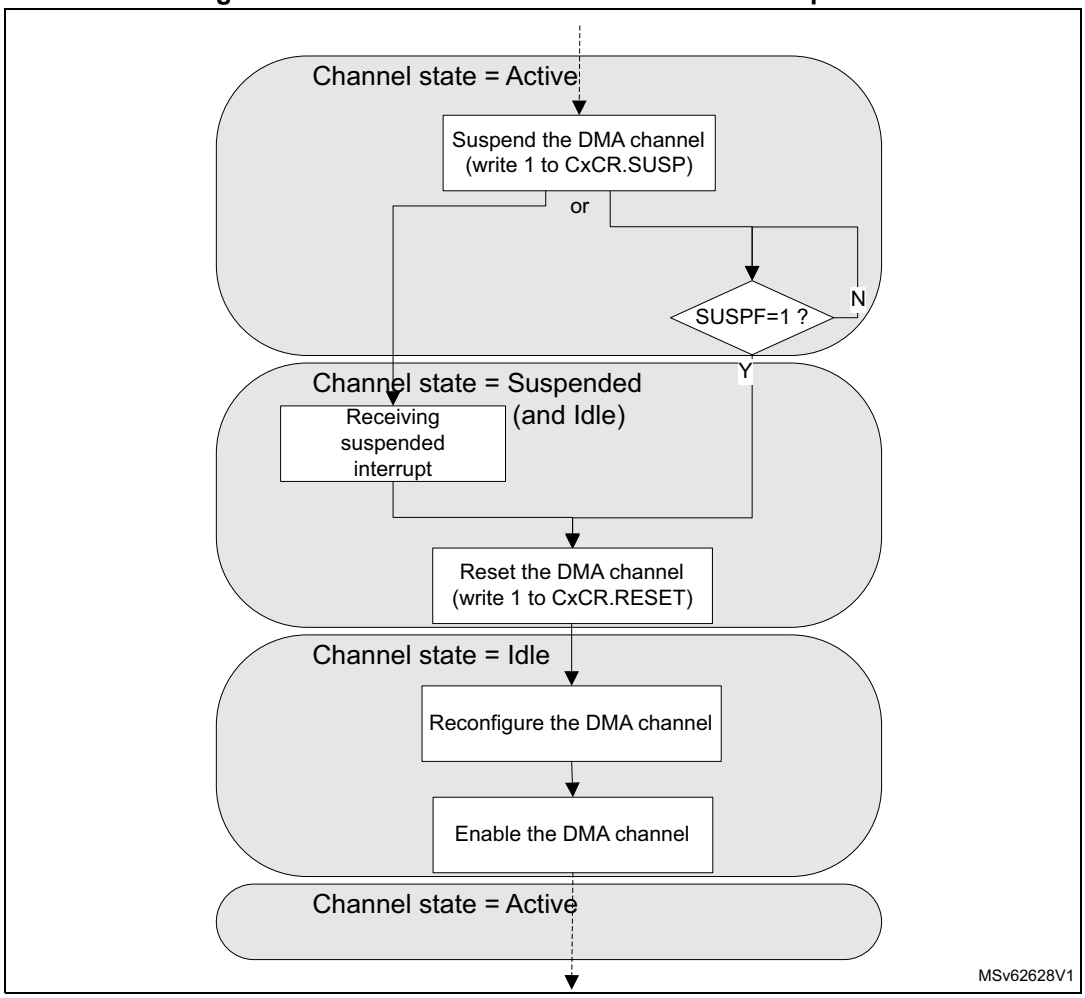
Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

17.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

Figure 116. GPDMA channel abort and restart sequence

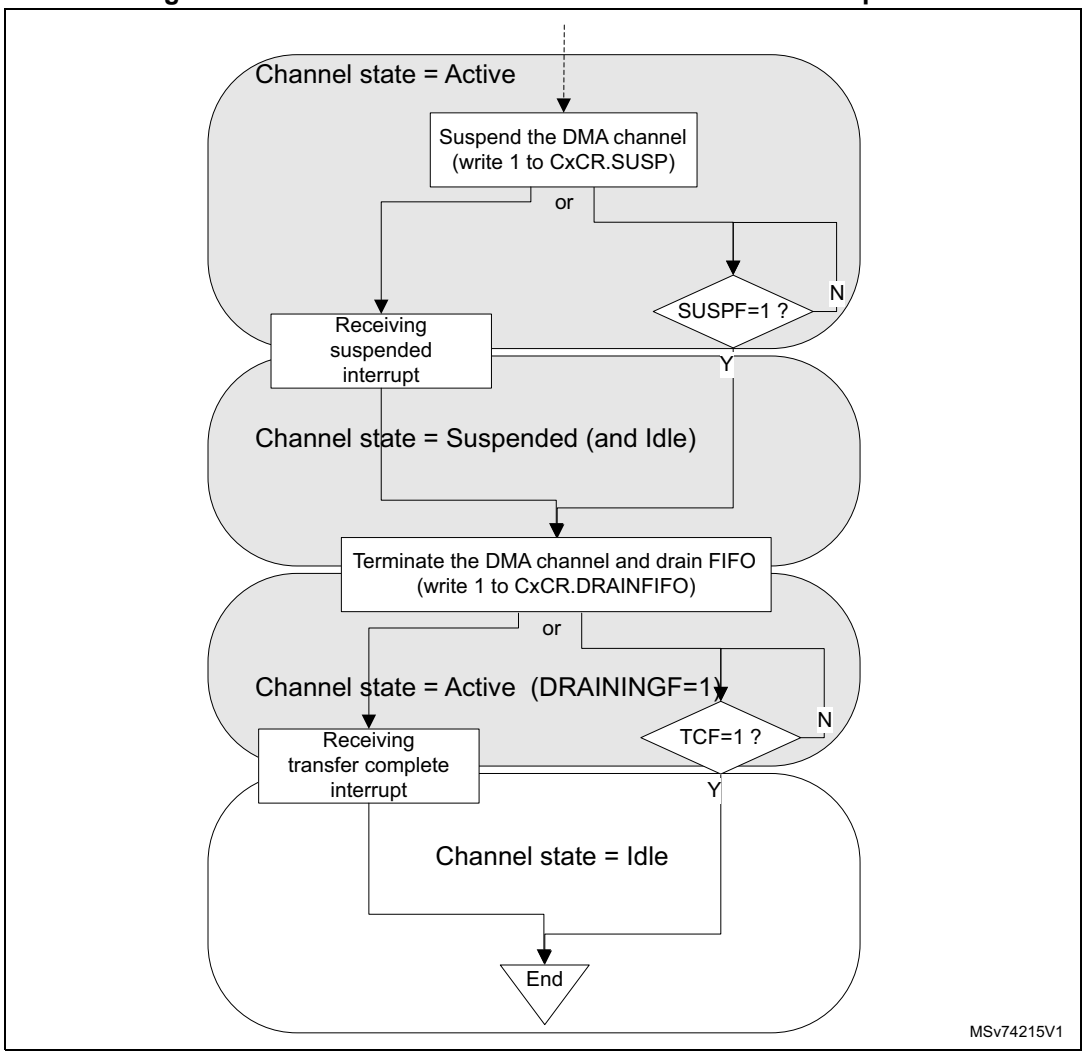


17.4.5 GPDMA channel terminate and drain FIFO

Alternatively, for terminating a continuous GPDMA transfer with a circular buffering or a double buffering, and having popped out any residual byte(s) in the channel FIFO and writing them to the destination, the software can terminate an active channel and drain the channel FIFO with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software orders to drain the channel FIFO by writing 1 to GPDMA_CxCR.DRAINFIFO.
4. The software polls transfer complete flag GPDMA_CxSR.TCF until TCF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.TCIE. Wait for the channel to be effectively with a drained FIFO means wait for completion of any residual write burst from the FIFO to the destination. Then the channel is disabled by hardware and in IDLE state.

Figure 117. GPDMA channel terminate and drain FIFO sequence



Note: If the softwares writes the GPDMA_CxCR register in a single shot with concurrently SUSP = 1, DRAINFIFO = 1, and RESET = 0, then the sequence for terminating and draining the FIFO also occurs.

GPDMA_CxTR2.DREQ = 1 and GPDMA_CxCR.DRAINFIFO = 1 is not supported, only peripheral-to-memory and memory-to-memory transfers are supported to be terminated with drained FIFO.

17.4.6 GPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the following UT1, UT2, UB1, USA, UDA, UT3 and UB2 if present, and ULL bits in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, GPDMA_CxTR3 and GPDMA_CxBR2 if present, and GPDMA_CxLLR) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Caution: The user must program the pointer to the next linked-list data structure (GPDMA_CxLLR[15:0]) not to exceed the 64-Kbyte addressable space defined by the link base address register (GPDMA_CxLBAR). The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA, plus UB2 and UT3 if present, and ULL) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 3) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 118*) and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory
- channel x (x = 4 to 5), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, GPDMA_CxTR3, GPDMA_CxBR2 and GPDMA_CxLLR, (see *Figure 119*), and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory

Figure 118. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

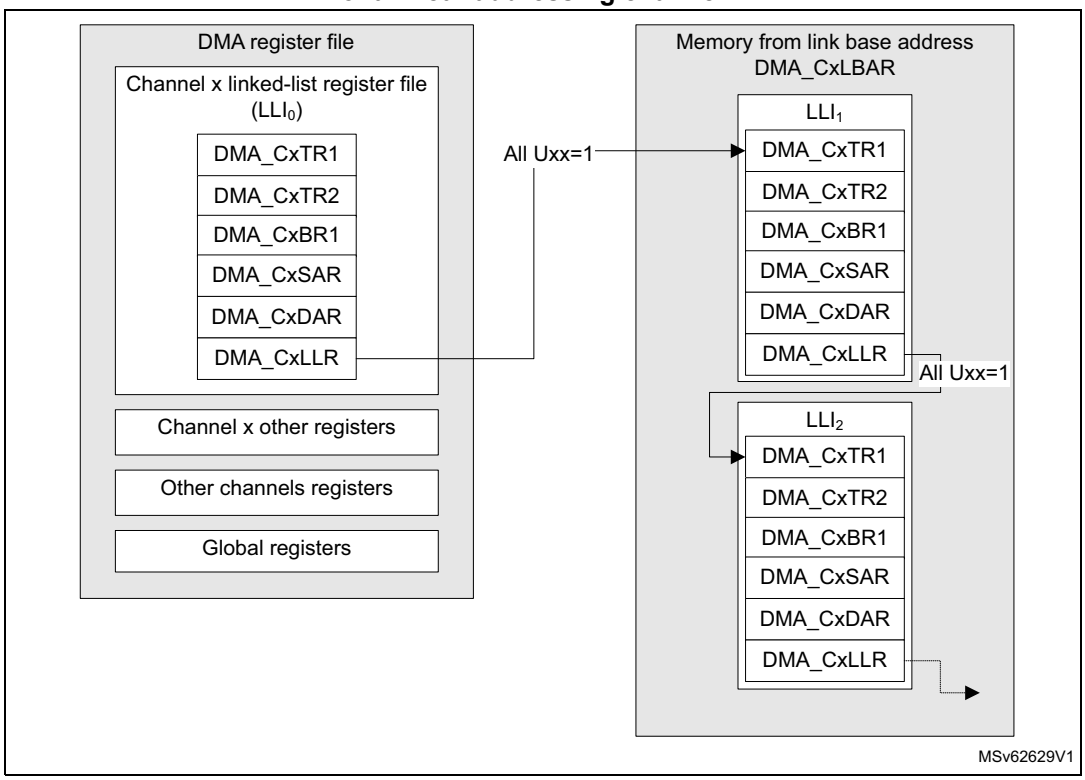
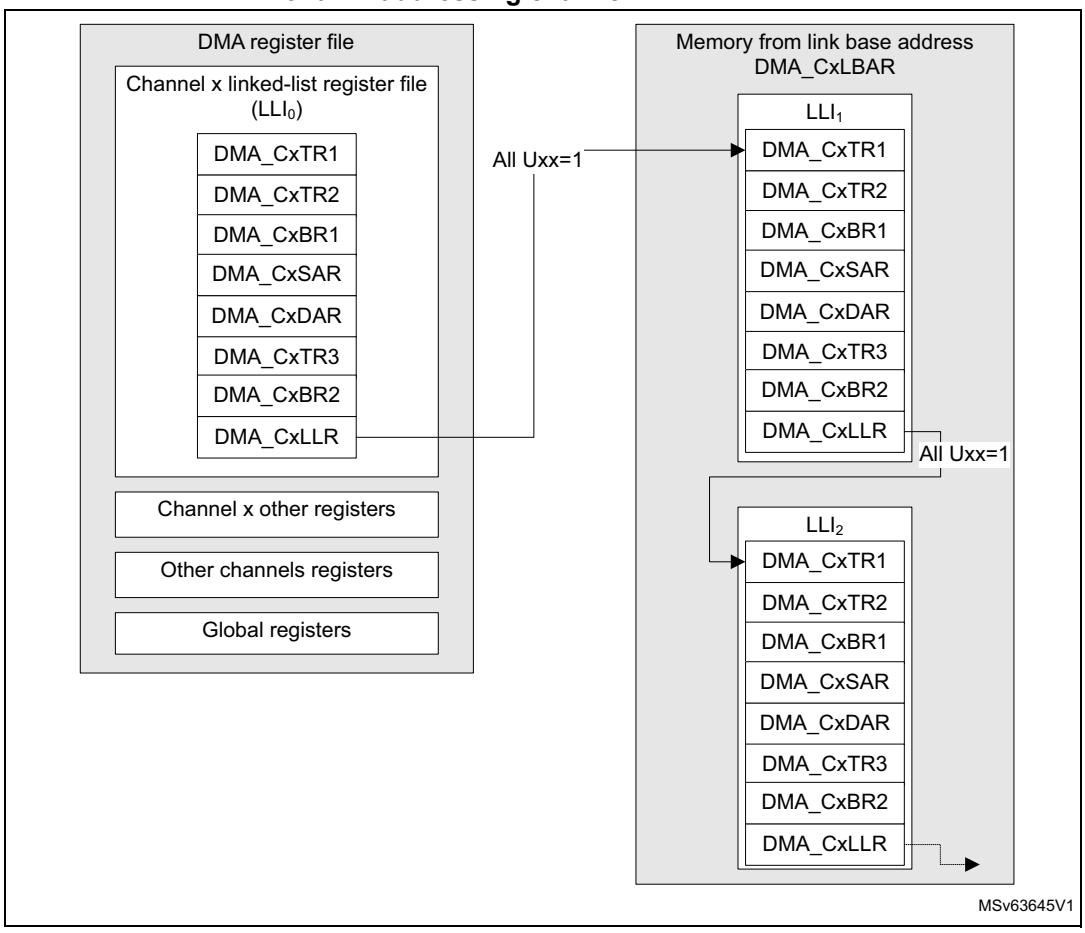


Figure 119. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see *Figure 120*).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in *Figure 121*.

Figure 120. GPDMA dynamic linked-list data structure of a linear addressing channel x

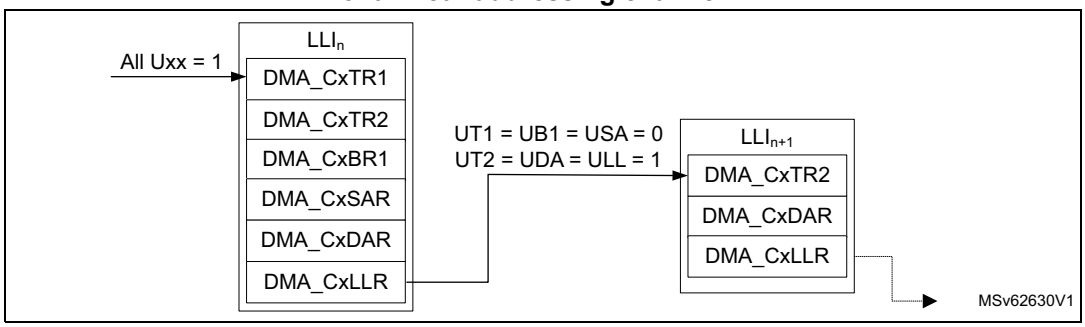
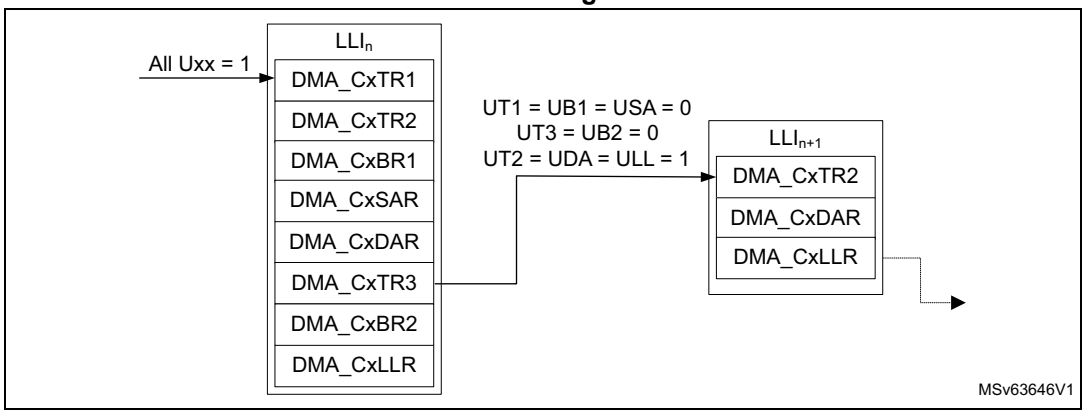


Figure 121. GPDMA dynamic linked-list data structure of a 2D addressing channel x



17.4.7 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the GPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.

17.4.8 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started

by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated with the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 17.4.15* for more details).

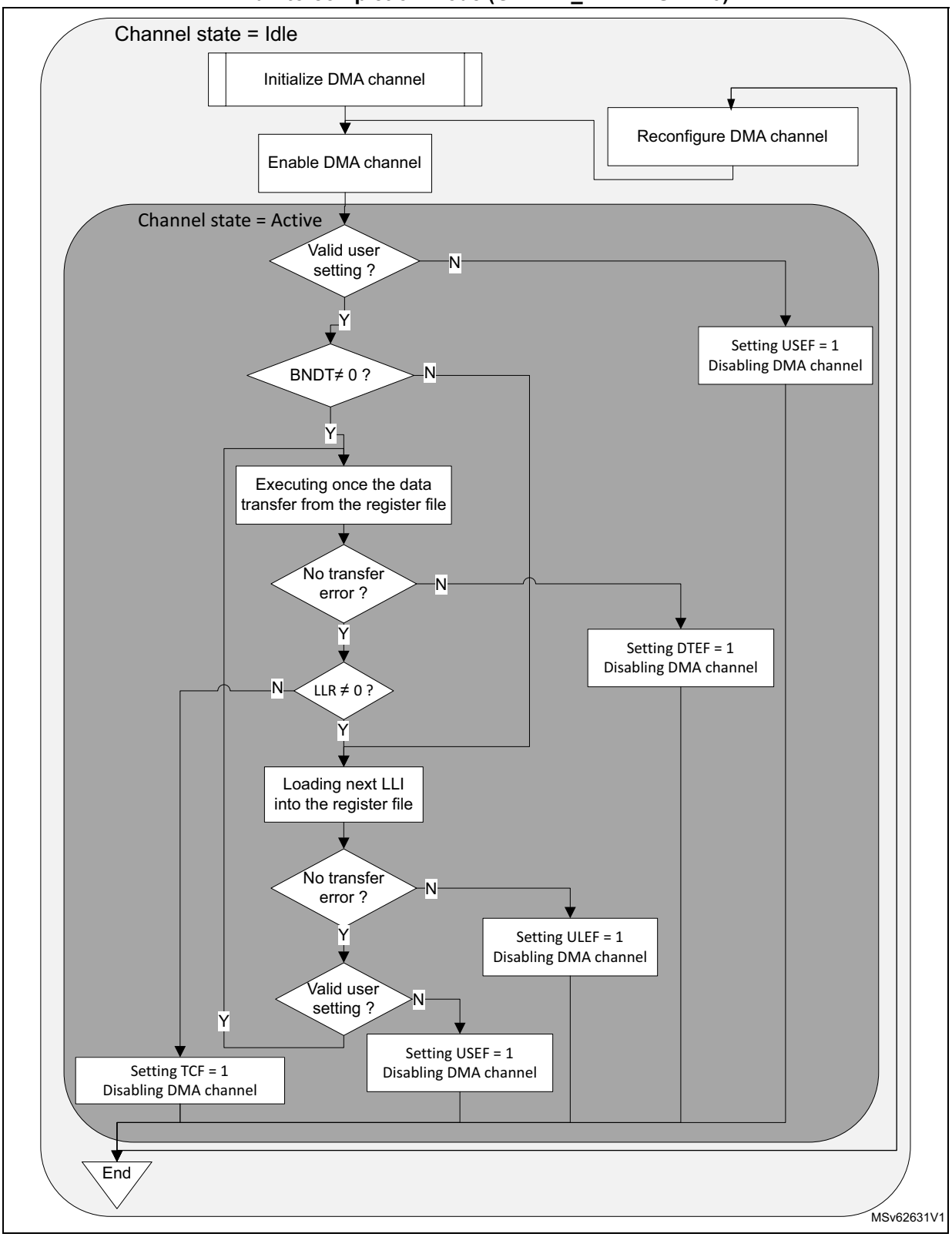
In typical run-to-completion mode, allocation of an GPDMA channel, including detailed programming, occurs once during GPDMA initialization. This process reserves a data communication link and GPDMA service during run-time for repeated transfers, such as transfers from a peripheral to memory, from memory to a peripheral, or memory-to-memory transfers. The reserved data communication link may consist of a dedicated channel, or the channel may be shared. A repeated transfer consists of a sequence of linked list items (LLIs).

Figure 122 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 122 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 122, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 17.3.6).

Figure 122. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

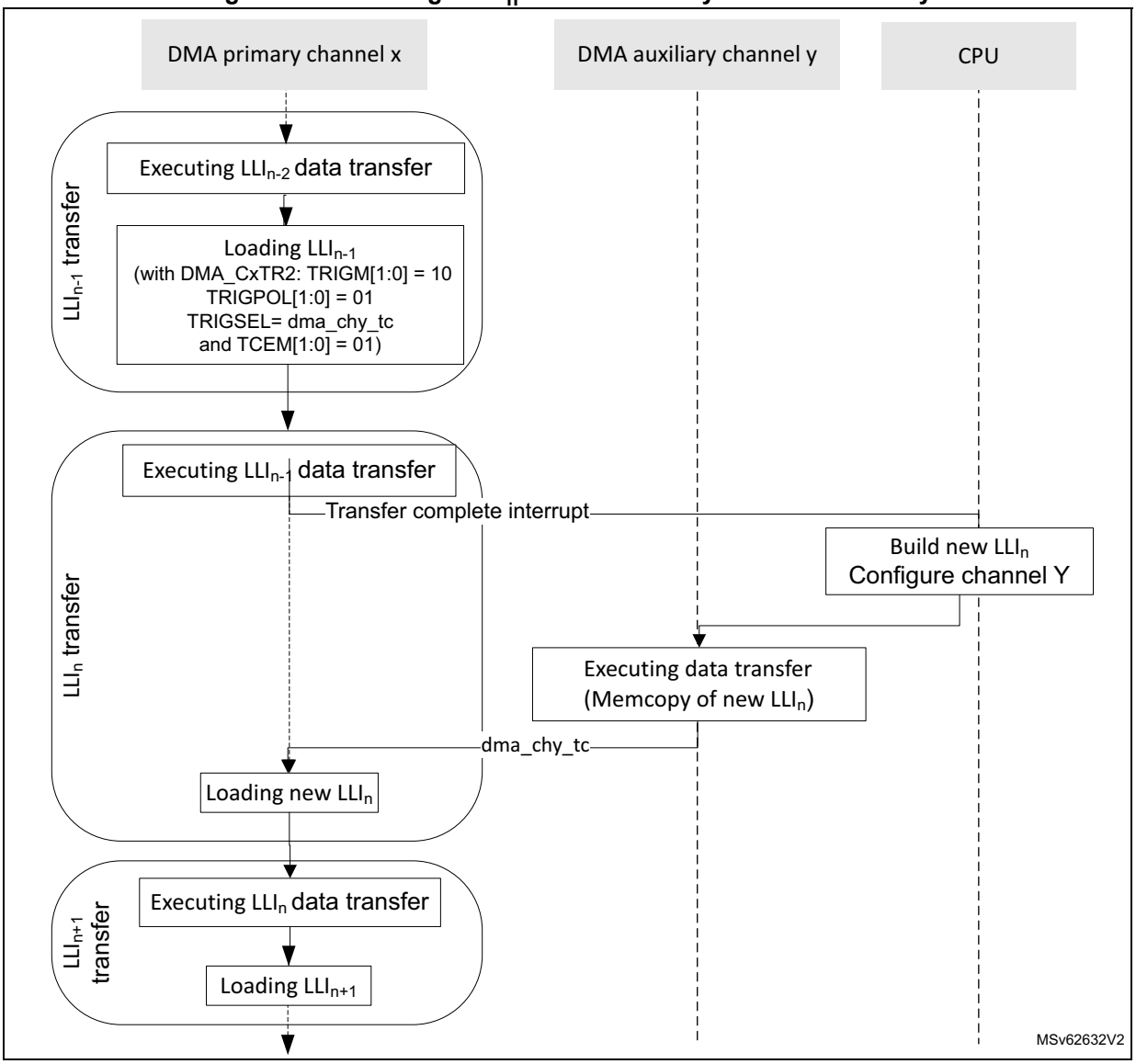
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 17.3.7* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 123 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 123. Inserting a LLI_n with an auxiliary GPDMA channel y



17.4.9 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 4 to 5), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.

  - GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
  - GPDMA_CxTR3 for channel x (x = 4 to 5) defines the source/destination additional address offset between burst transfers.
  - GPDMA_CxBR2 for channel x (x = 4 to 5) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated with the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 17.4.15* for more details).

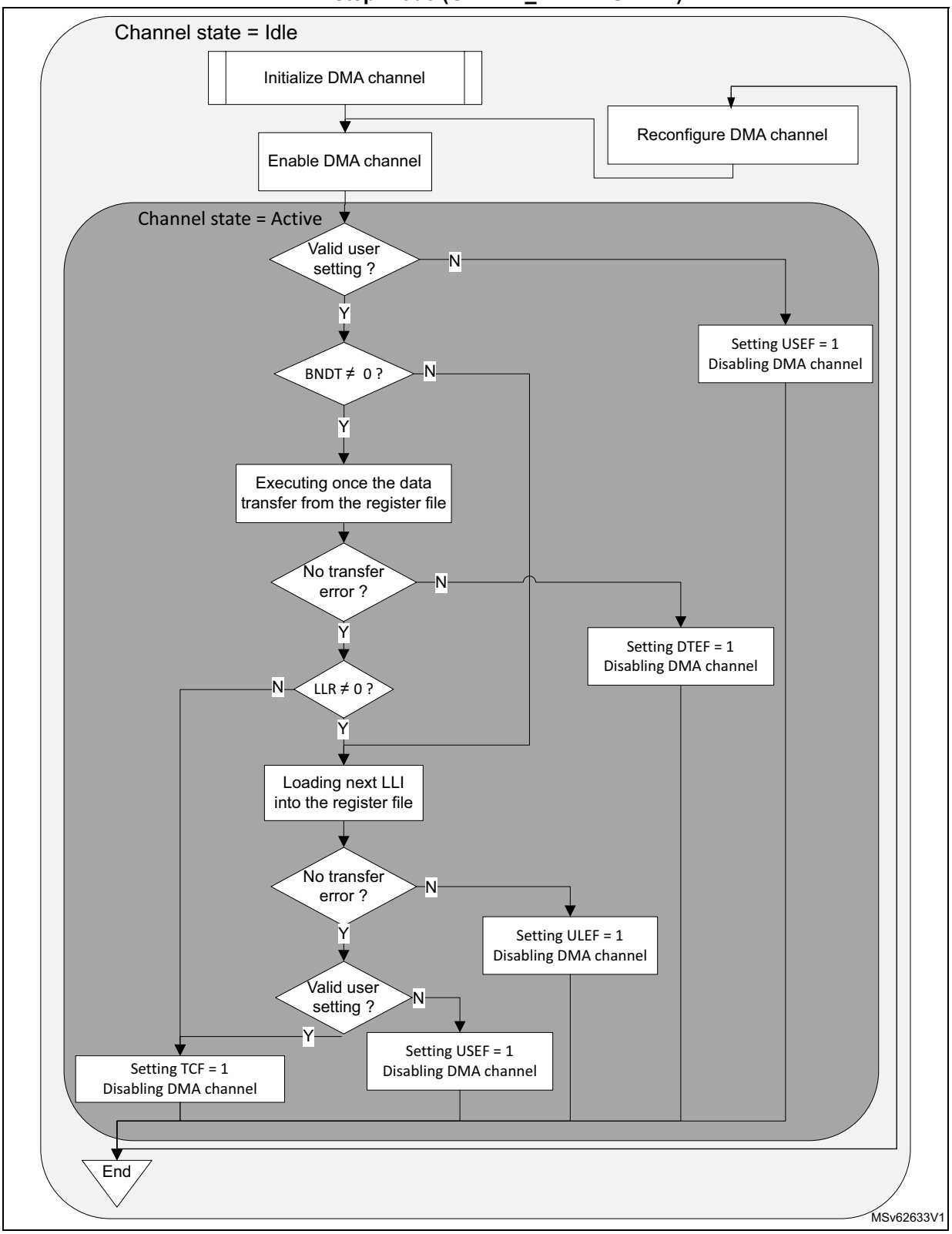
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 124 depicts the GPDMA channel execution mode, and its programming in link step mode.

Note: Figure 124 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 124, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 17.3.6).

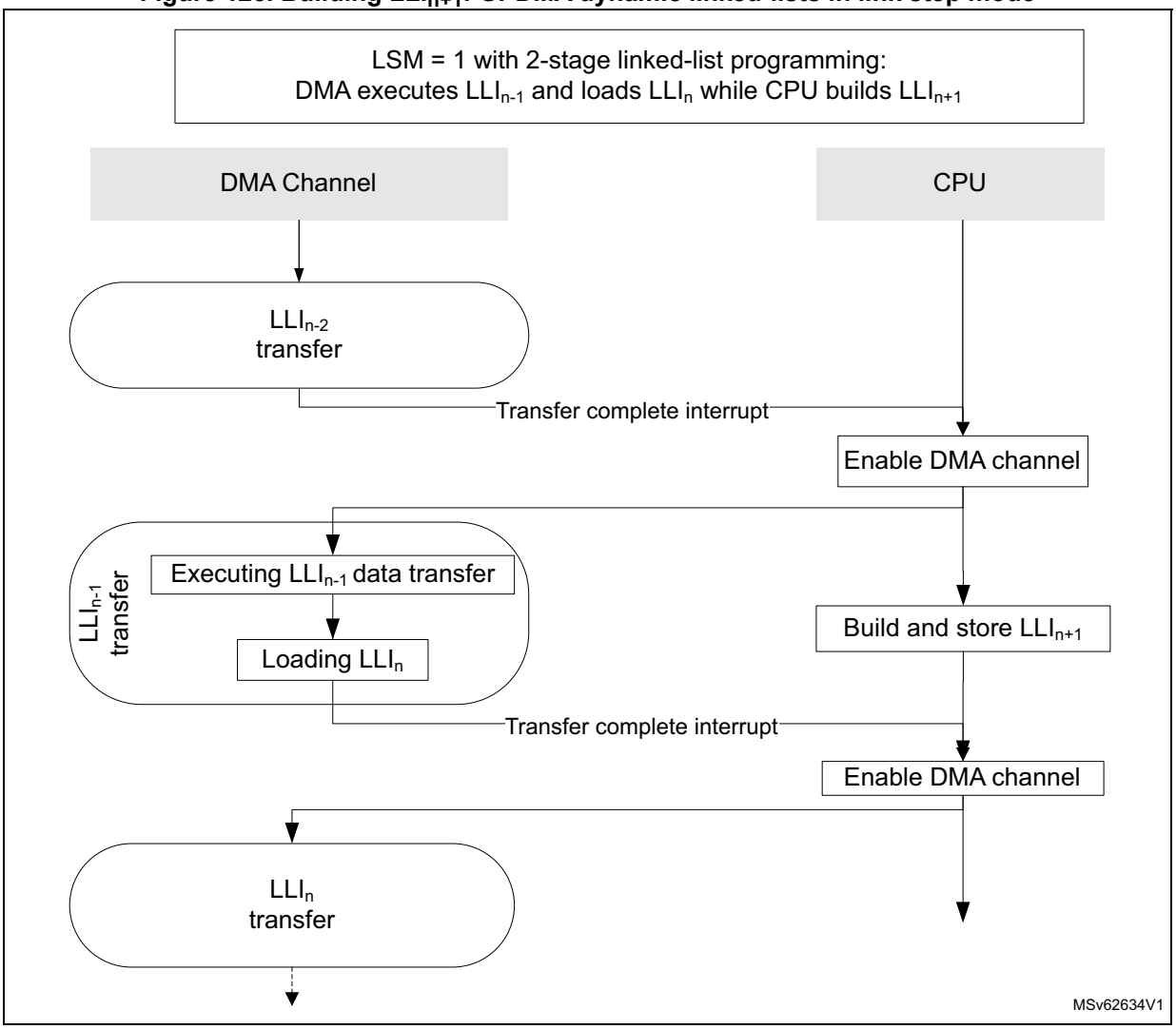
Figure 124. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in *Figure 125*.

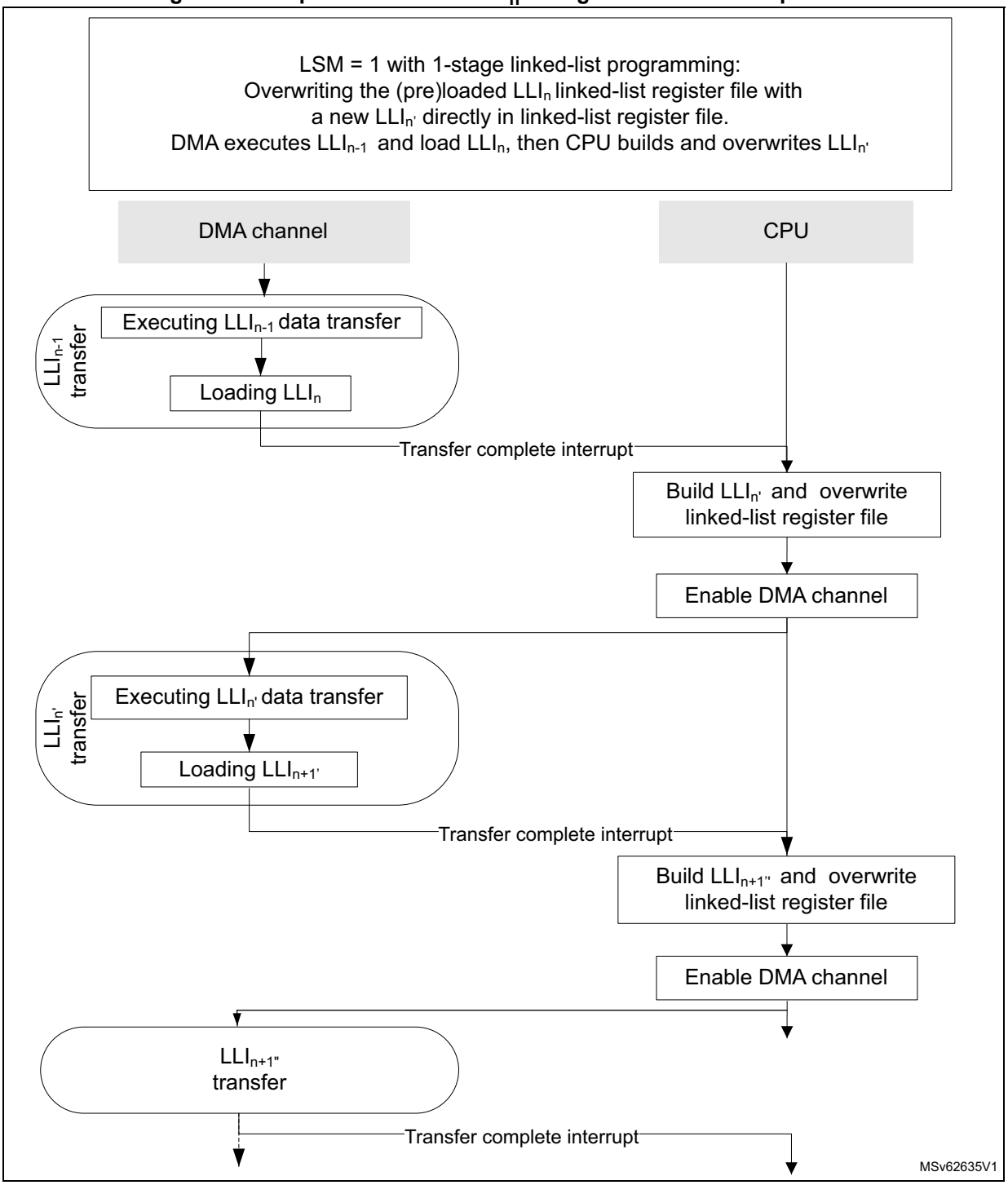
Figure 125. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in *Figure 126*.

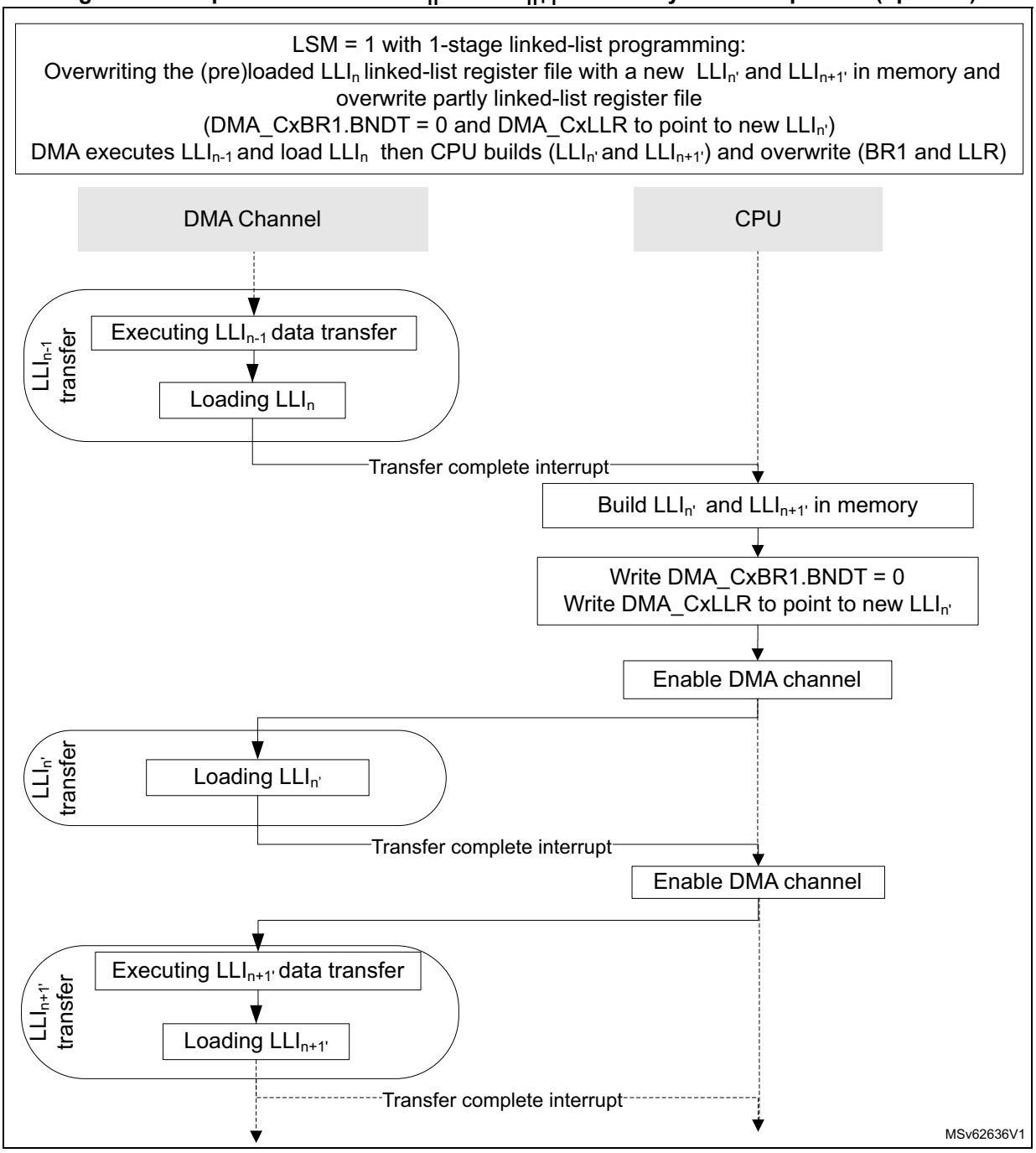
Figure 126. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n')}$ as shown in *Figure 127*.

Figure 127. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



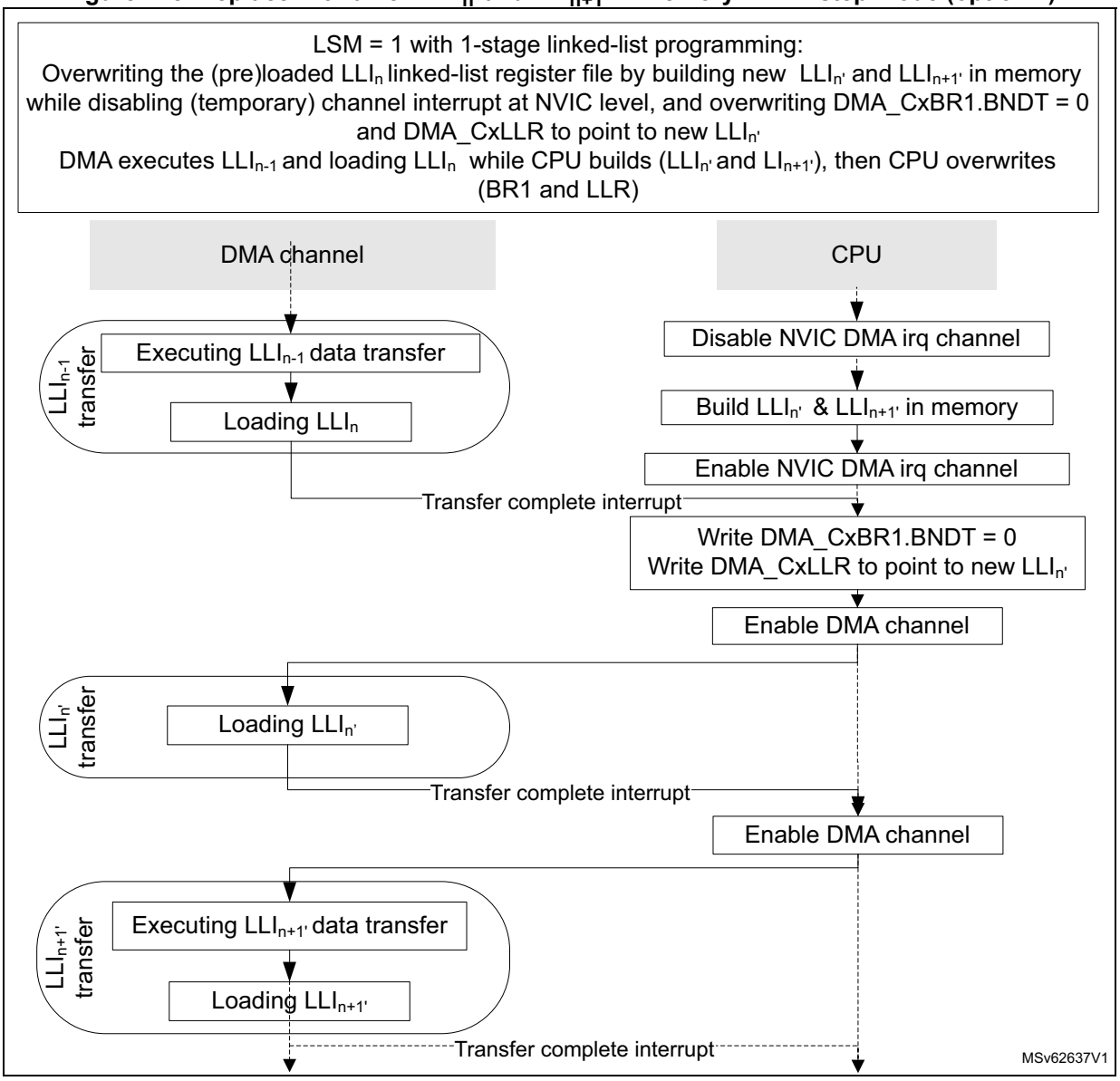
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile GPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 128*.

Figure 128. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



17.4.10 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the GPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 17.4.4*.)

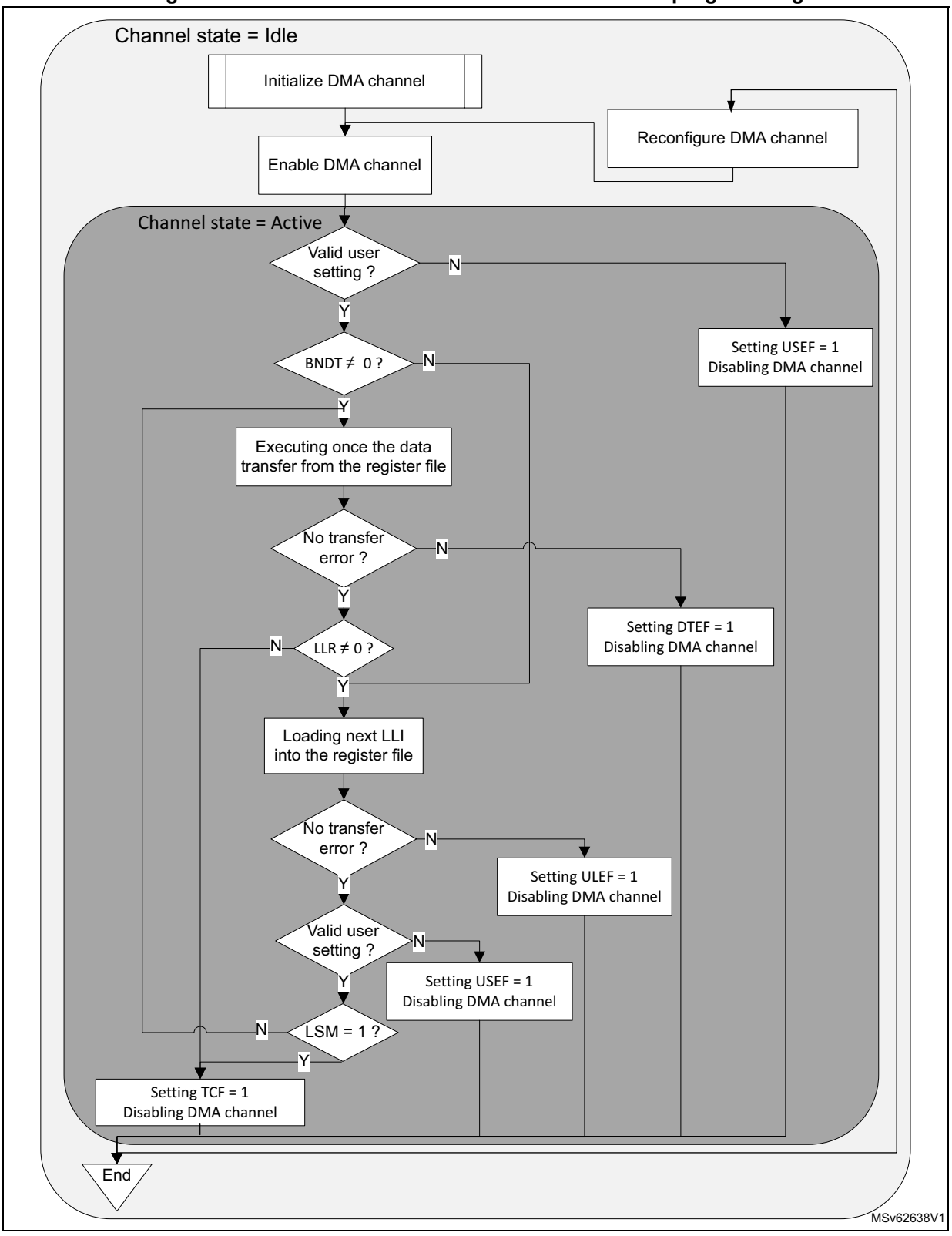
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 129 shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Note: Figure 129 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 129, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 17.3.6).

Figure 129. GPDMA channel execution and linked-list programming



17.4.11 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 17.3.2* for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

Table 168. Programmed GPDMA source/destination burst

SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/beats)	Next data/ beat address	Next burst address	Burst address alignment
00	1	0 (fixed)	n = 0 to 63(1)	n+1	+ 0	+ 0	1
01	2						2
10	4						4
00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
01	2				+ 2	+ 2 * (n + 1)	2
10	4				+ 4	+ 4 * (n + 1)	4
11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

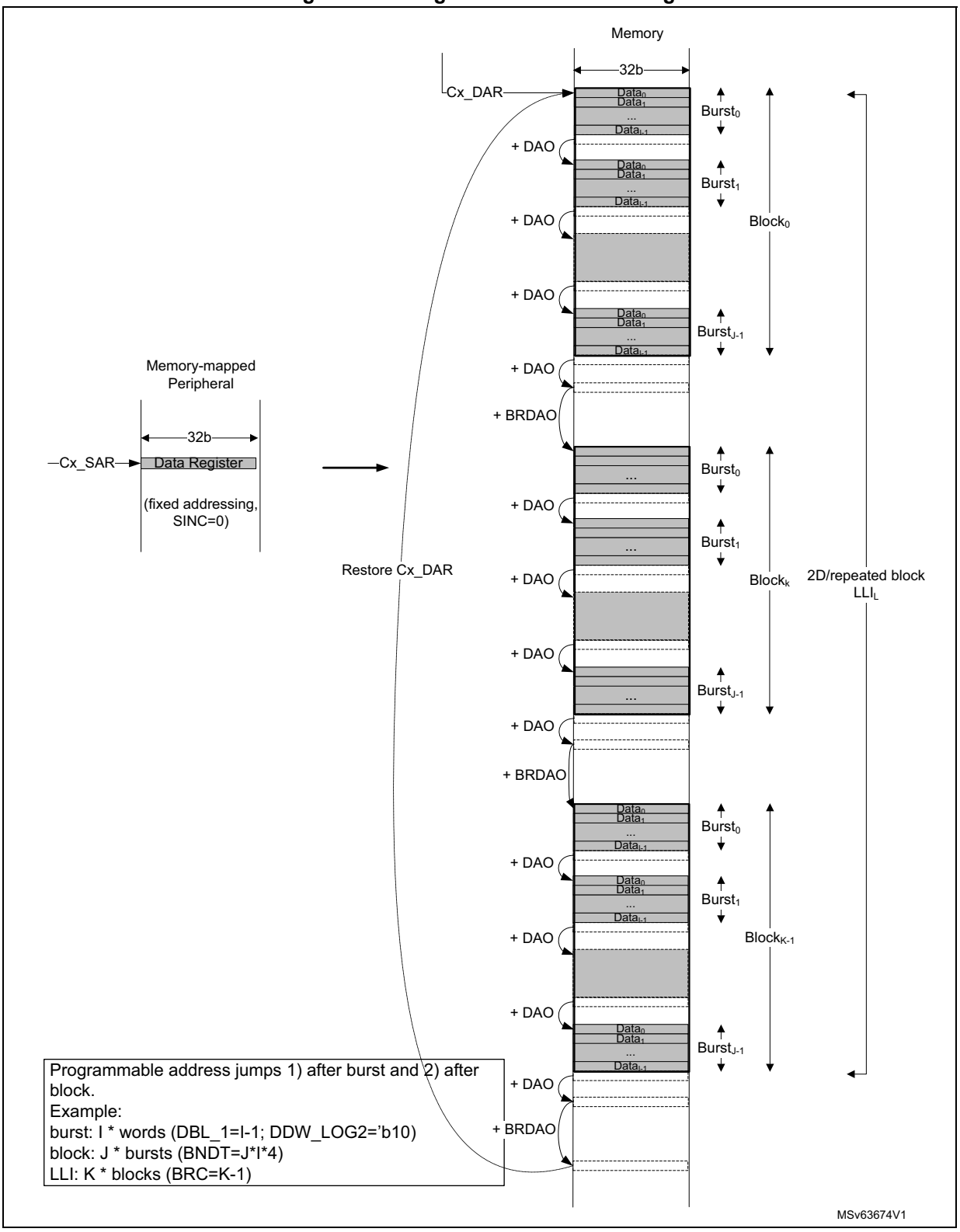
GPDMA burst with 2D addressing (channel x = 4 to 5)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 130* with a 2D destination buffer.

Figure 130. Programmed 2D addressing



GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is the half the FIFO size of channel x, in bytes (see *Section 17.3.2* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size. Each transfer is of a size that is lower than or equal to half of the FIFO size, without any user constraint.
- If the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields (see *Table 169*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 169. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)	0		$0B_3,0B_2,0B_1,0B_0$
							1		B_30,B_20,B_10,B_00
						01 (RA, SE)	0		SB_3,SB_2,SB_1,SB_0
							1		B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$000B_1,000B_0$
							1		$00B_10,00B_00$
							0	1	$0B_100,0B_000$
							1		B_1000,B_0000
						01 (RA, SE)	0	0	$SSSB_1,SSSB_0$
							1		SSB_1S,SSB_0S
							0	1	SB_1SS,SB_0SS
							1		B_1SSS,B_0SSS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$		00	Byte	00 (RA, LT)	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

Table 169. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	x	01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$00B_3B_2,00B_1B_0$
							1		$00B_2B_3,00B_0B_1$
							0	1	B_3B_200,B_1B_000
							1		B_2B_300,B_0B_100
						01 (RA, SE)	0	0	SSB_3B_2,SSB_1B_0
							1		SSB_2B_3,SSB_0B_1
							0	1	B_3B_2SS,B_1B_0SS
							1		B_2B_3SS,B_0B_1SS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						10 (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)	0		B_5B_4,B_1B_0
							1		B_4B_5,B_0B_1
						01 (LA, RT)	0		B_7B_6,B_3B_2
							1		B_6B_7,B_2B_3
						1x (UNPACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$

Table 169. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	10	Word	xx	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
			1	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						1x (UNPACK)			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)	0		B_6B_4,B_2B_0
							1		B_4B_6,B_0B_2
						01 (LA, RT)	0		B_7B_5,B_3B_1
							1		B_5B_7,B_1B_3
						1x (UNPACK)	0		$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1		$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0	$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7,B_0B_2B_1B_3$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

17.4.12 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[7:0] field (see *Section 17.3.4*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 131*, is defined by:

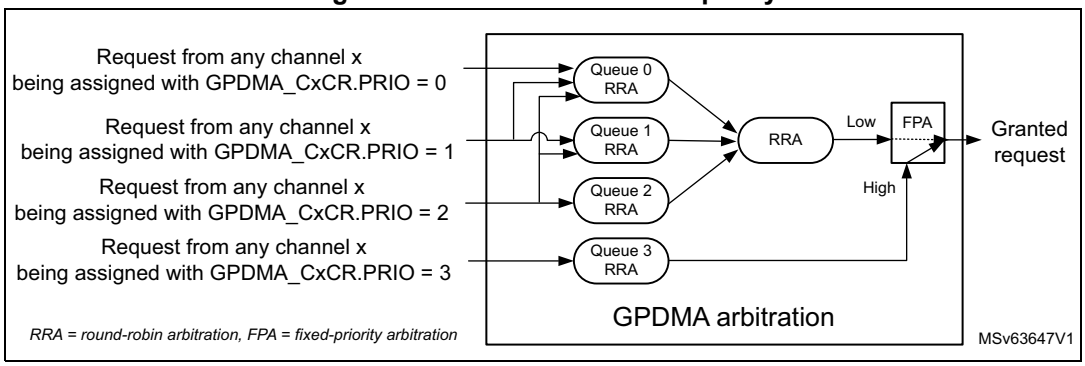
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1, or 2) for non time-sensitive channels with priority 0, 1, or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.

– Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
– Any queue 0, 1, or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
– Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1, and queue 2.

Figure 131. GPDMA arbitration policy



GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  – If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).
  $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).
  $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$
  $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).
  $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$
  $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$
  The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.

- **Example 2**: application where the user distributes a same non-null N number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is
    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is
    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is
    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is
    $B_{n, P0} = 1/N *1/9 * (1 - B_{Q3})$
    $B_{n, P1} = 1/N *5/18 * (1 - B_{Q3})$
    $B_{n, P2} = 1/N *11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for $J > 0$ (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ ($K>0$ in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 ($K > 0$)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 ($K = 0$)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

17.4.13 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see *Section 17.3.7* for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel x (x = 4 to 5), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in GPDMA_CxTR2 for more details).

Trigger hit memorization and trigger overrun flag generation

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

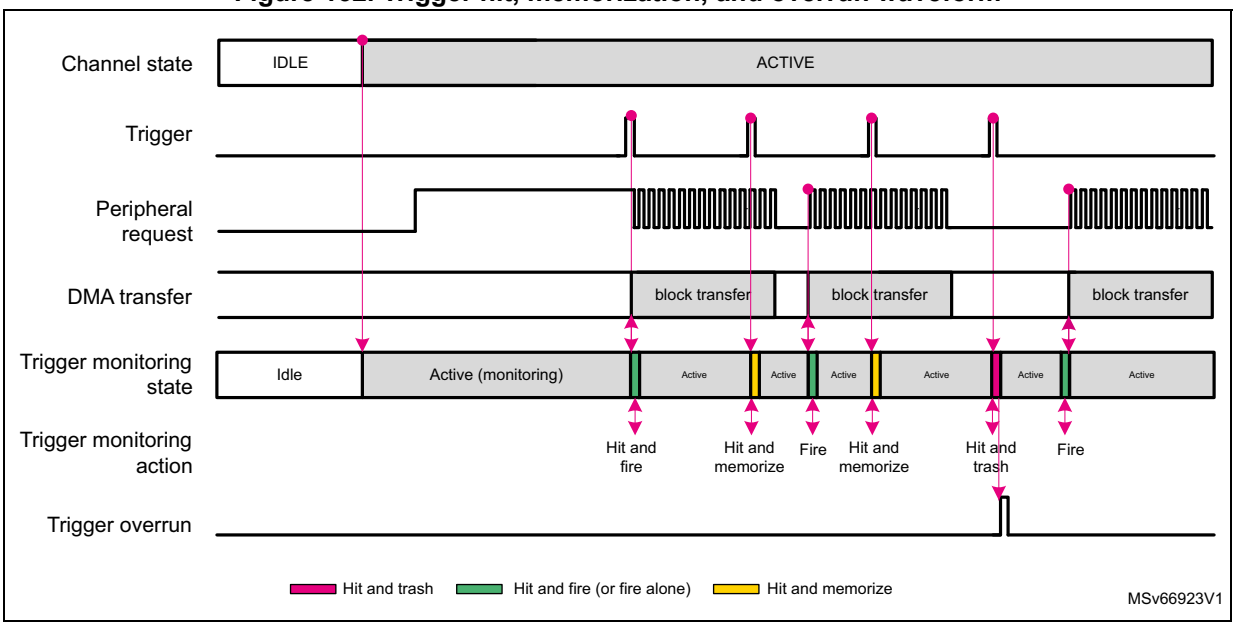
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Figure 132 illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 132. Trigger hit, memorization, and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

17.4.14 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

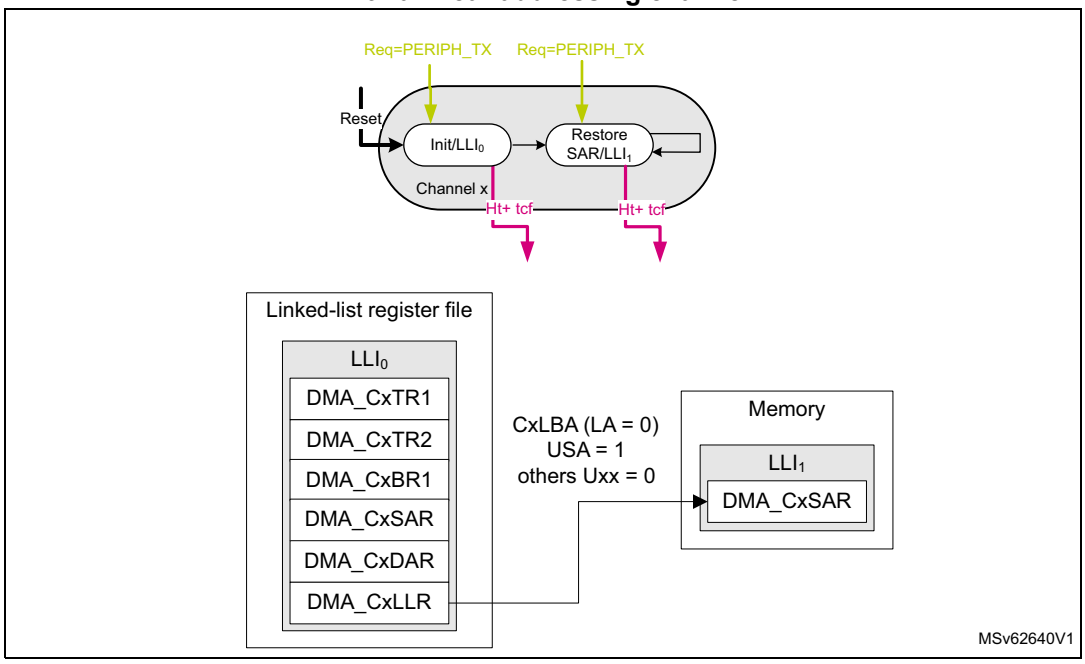
For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete transfer

events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

Figure 133 illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

Figure 133. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel

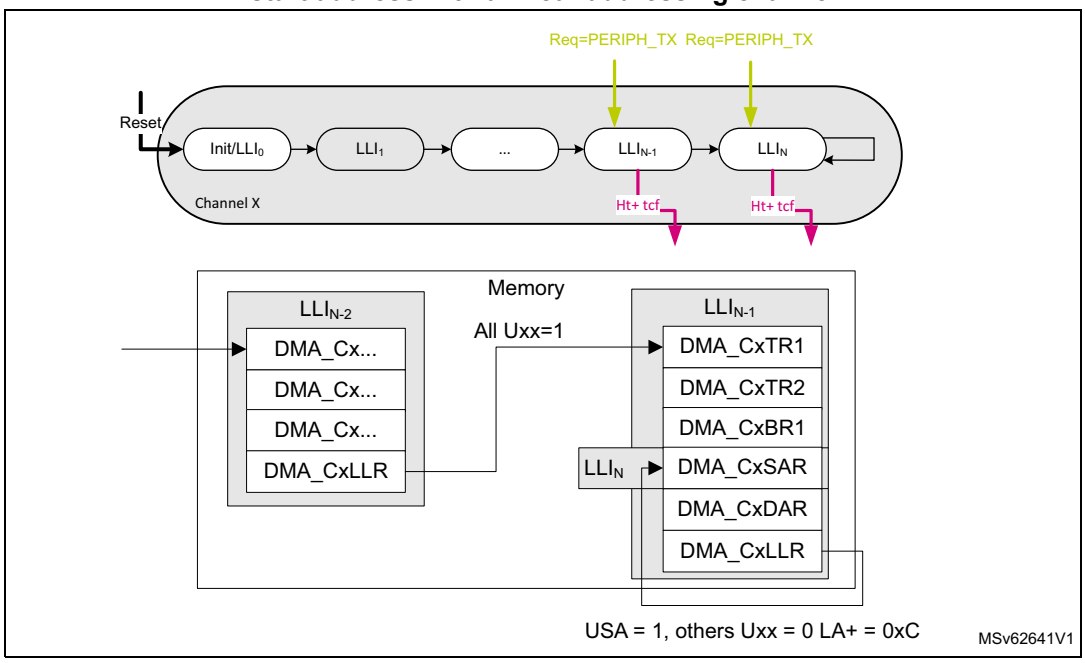


Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

Figure 134 illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 134. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



17.4.15 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 17.3.6*).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with a hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

17.4.16 GPDMA byte-based padding at the block level

For a given channel x, a LLI_n data transfer consists of a block transfer and is defined by the source block size GPDMA_CxBR1.BNDT[15:0], the size of the block of data to be transfered from the source, in bytes.

During a peripheral-to-memory transfer, a peripheral may be streaming and receiving a given number of bytes (for example, via an I2C, UART/USART/LPUART, ADC, DAC) or half-words, and this block size may not be a multiple of a word or a double-word. It is important to allow the user to program any number of bytes for the GPDMA source block size independently from the source data width and the destination data width, in order to use efficiently the bandwidth of the 32-bit AHB system bus and/or the 64-bit AXI system bus when accessing and writing into memory (embedded SRAM, NVM or external memory). This enables to decrease the bus bandwidth for the memory transfers by a ratio of up to 4 in AHB and up to 8 in AXI. More generally, having the source block size independent from the source data width and the destination data width can be useful also with memory-to-peripheral transfers, and with memory-to-memory transfers (AHB from/to AXI).

A FIFO-based channel is able to perform byte-based padding on the last write(s) up to the fully transferred source block size. Then channel FIFO is empty. Data handling is performed on zero bytes in the same way as on regular data bytes from the source, as explained in *GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment*.

There is byte-based padding at the block level:

1. If source data width is less than the destination data width, and if FIFO queuing and data packing is configured (PAM[1:0] = 1x):

- Padding with zero bytes. For example, if BNDT= 5, 8-bit reads, 32-bit writes:
  - if DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 000B_4, $B_3B_2B_1B_0$
  - if DBX = 0, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> 0$B_4$00, $B_1B_0B_3B_2$
  - if DBX = 1, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> 00$B_4$0, $B_2B_3B_0B_1$
  - if DBX = 1, DHX = 1: B_4,B_3,B_2,B_1,B_0 -> $B_4$000, $B_0B_1B_2B_3$

2. If source data width > destination data width, and if FIFO queuing and data unpacking is configured (PAM[1:0] = 1x):

- Padding with zero bytes.
- On the last read, only the number of LSB(s) up to the block size are written to the FIFO. The last written data to the destination is padded with zero bytes up to the destination data width. For example, if BNDT= 5, 32-bit reads, 16-bit writes:
  - if DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 0B_4, B_3B_2, B_1B_0
  - if DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$, -> $B_4$0, B_2B_3, B_0B_1

3. If source data width = destination data width (and BNDT[15:0] is not a multiple of that value):

- Padding with zero bytes. For example, if BNDT= 6, 32-bit reads, 32-bit writes are configured
  - if SBX = 0, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_5B_4$, $B_3B_2B_1B_0$
  - if SBX = 0, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_5B_400, $B_1B_0B_3B_2$
  - if SBX = 0, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_4B_5$, $B_2B_3B_0B_1$
  - if SBX = 0, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4B_500, $B_0B_1B_2B_3$
  - if SBX = 1, DBX = 0, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $000B_4$, $B_3B_1B_2B_0$
  - if SBX = 1, DBX = 0, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_400$, $B_2B_0B_3B_1$
  - if SBX = 1, DBX = 1, DHX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $00B_40$, $B_1B_3B_0B_2$
  - if SBX = 1, DBX = 1, DHX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_4000, $B_0B_2B_1B_3$

4. If source data width < destination data width, and if destination data is configured as right-aligned and, either configured with zero padding bytes (PAM[1:0] = 00), or configured with sign extension (PAM[1:0] = 01):

- Padding with zero bytes if PAM[1:0] = 00. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 00:
  - if DBX =0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $000B_4$, $000B_3$, $000B_2$, $000B_1$, $000B_0$
- Padding with sign extension if PAM[1:0] = 01. For example, BNDT = 5, 8-bit reads, 32-bit writes, PAM[1:0] = 01:
  - if DBX = 0, DHX = 0: B_4,B_3,B_2,B_1,B_0 -> $SSSB_4$, $SSSB_3$, $SSSB_2$, $SSSB_1$, $SSSB_0$

5. If source data width > destination data width, and if destination data is configured as right-aligned (PAM[1:0] = 00), or left-aligned (PAM[1:0] = 01):

- Padding with zero bytes. For example, BNDT = 5, 32-bit reads, 16-bit writes:
  - if SBX = 0, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_1B_0
  - if SBX = 0, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_1
  - if SBX = 0, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_2
  - if SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_2B_3
  - if SBX = 1, PAM[1:0] = 00, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> $0B_4$, B_2B_0
  - If SBX = 1, PAM[1:0] = 00, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> B_40, B_0B_2
  - If SBX = 1, PAM[1:0] = 01, DBX = 0: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_3B_1
  - If SBX = 0, PAM[1:0] = 01, DBX = 1: $B_7B_6B_5B_4$, $B_3B_2B_1B_0$ -> 00, B_1B_3

17.4.17 GPDMA secure/nonsecure channel

The GPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel x is a secure or a nonsecure hardware resource, as configured by GPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A nonsecure read access to a register field of this channel is forced to return 0, except for GPDMA_SECCFGR, GPDMA_PRIVCFGR, GPDMA_RCFGLOCKR that are readable by a nonsecure agent.
- A nonsecure write access to a register field of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or nonsecure the GPDMA data transfer from the source (GPDMA_CxTR1.SSEC) and the GPDMA data transfer to the destination (GPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with secure transfers via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a secure bus that reflects GPDMA_SECCFGR, to keep the other peripherals informed of the secure/nonsecure state of each GPDMA channel x.

The GPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure GPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transfer as a nonsecure software is not allowed to do so and must have GPDMA_CxCR.EN = 0 before the nonsecure software can reprogram the GPDMA_CxCR for a next transfer. The secure software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

17.4.18 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for GPDMA_PRIVCFGR, GPDMA_SECCFGR, GPDMA_RCFGLOCKR that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the

privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

17.4.19 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single or a burst) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a GPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).

- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 4 to 5): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 4 to 5): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 4 to 5): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 4 to 5): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1. BNDT = 0)

17.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

17.6 GPDMA in low-power modes

Table 170. Effect of low-power modes on GPDMA

Mode	Description
Sleep	No effect. GPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the GPDMA registers is kept when entering Stop mode.
Standby	The GPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 17.3.3* to know if any Stop mode is supported.

17.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 171. GPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
GPDMA_CHx	Transfer complete	GPDMA_CxCR.TCIE	GPDMA_CxSR.TCF	Write 1 to GPDMA_CxFCR.TCF
	Half transfer	GPDMA_CxCR.HTIE	GPDMA_CxSR.HTF	Write 1 to GPDMA_CxFCR.HTF
	Data transfer error	GPDMA_CxCR.DTEIE	GPDMA_CxSR.DTEF	Write 1 to GPDMA_CxFCR.DTEF
	Update link error	GPDMA_CxCR.ULEIE	GPDMA_CxSR.ULEF	Write 1 to GPDMA_CxFCR.ULEF
	User setting error	GPDMA_CxCR.USEIE	GPDMA_CxSR.USEF	Write 1 to GPDMA_CxFCR.USEF
	Suspended	GPDMA_CxCR.SUSPIE	GPDMA_CxSR.SUSPF	Write 1 to GPDMA_CxFCR.SUSPF
	Trigger overrun	GPDMA_CxCR.TOFIE	GPDMA_CxSR.TOF	Write 1 to GPDMA_CxFCR.TOF

A GPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

- A half-transfer event is a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see *Section 17.4.19*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

17.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

17.8.1 GPDMA secure configuration register (GPDMA_SECCFGR)

Address offset: 0x00

Reset value: 0x0000 0000

A write access to this register must be secure and privileged. A read access is secure or nonsecure, privileged or unprivileged.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (when GPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **SECx**: secure state of channel x (x = 5 to 0)
0: Nonsecure
1: Secure

17.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure.

This register can mix secure and nonsecure information. If a channel x is configured as secure (GPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN =1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **PRIVx**: privileged state of channel x (x = 5 to 0)
0: unprivileged
1: privileged

17.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attributes in order to individually lock, for example at boot time, the secure privileged attributes of any GPDMA

channel/resource (to lock the setting of GPDMA_CxSECCFGR, GPDMA_CxPRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure.

Note: If TZEN = 0, this register cannot be written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
										rs	rs	rs	rs	rs	rs

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **LOCKx**: lock the configuration of GPDMA_SECCFGR.SECx, GPDMA_PRIVCFGR.PRIVx until a global GPDMA reset (x = 5 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

17.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with GPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of GPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of GPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in GPDMA_CxFCR.

If a channel x is in secure state (GPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **MISx**: masked interrupt status of channel x (x = 5 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

17.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR)

Address offset: 0x10

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (GPDMA_SECCFGR.SECx = 1). It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **MISx**: masked interrupt status of the secure channel x (x = 5 to 0)
0: no interrupt occurred on the secure channel x
1: an interrupt occurred on the secure channel x

17.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (GPDMA_SECCFGR.SECx).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LBA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **LBA[31:16]**: linked-list base address of GPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

17.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C+ 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: corresponding TOF flag cleared

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half-transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.

17.8.8 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAINING F	IDLEF
	r	r	r	r	r	r	r							r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level (in bytes)
Number of available bytes in the FIFO.
Note: After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag
0: no trigger overrun event
1: a trigger overrun event occurred

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half-transfer flag
0: no half-transfer event
1: a half-transfer event occurred
A half-transfer event is either a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).
A half-block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.
A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2)) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DRAININGF**: FIFO draining flag
0: No received order to drain the FIFO or channel FIFO draining completed
1: Ongoing channel FIFO draining and channel termination
This flag is asserted after that the order to drain the FIFO is received, and meanwhile that the channel is enabled and draining the FIFO and before the channel is terminated with a drained FIFO.
This flag is deasserted by hardware when the enabled channel has been ordered to be terminated with FIFO draining, and the hardware effectively terminates.

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is deasserted by hardware when the channel is enabled (GPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

17.8.9 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM
								rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAIN FIFO	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw					w	rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** priority level of the channel x GPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: linked-list allocated port
This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed **once** for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half-transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DRAINFIFO:** empty FIFO
This bit is write only. Writing 0 has no impact.
Writing 1 to RESET in this register causes the hardware to deassert this DRAINFIFO bit, whatever is written into this DRAINFIFO. Else:
Software must write 1 in order to drain the channel FIFO on a suspended channel.
If the software concurrently writes 1 to DRAINFIFO and 1 to SUSP and 0 to reset, the hardware suspends the channel and drains the FIFO by popping out the residual FIFO bytes to the destination (see the programming sequence in *Figure 117*).
0: Write: no action
1: Write: drain FIFO after having suspended the channel

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 115*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset

This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP, DRAINFIFO and EN bits (whatever is written receptively in bit 3 and bit 2 and bit 0).

The reset is effective when the channel is in steady state, meaning one of the following:

- active channel in suspended state (GPDMA_CxSR.SUSPF = 1, GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1 and GPDMA_CxSR.DRAININGF = 0)
- channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).

After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 116*).

0: no channel reset

1: channel reset

Bit 0 **EN:** enable

Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:

this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: write: ignored, read: channel disabled

1: write: enable channel, read: channel enabled

17.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or unprivileged, depending on the privileged state of the channel x in GPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **DSEC**: security attribute of the GPDMA transfer to the destination

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer to the destination is nonsecure.

Bit 30 **DAP**: destination allocated port

This bit is used to allocate the master port for the destination transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **DHX**: destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word:

0: no halfword-based exchanged within word

1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0:no byte-based exchange within half-word

1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: destination incrementing burst

0: fixed burst

1: contiguously incremented burst

The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and if present address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

Bit 15 **SSEC**: security attribute of the GPDMA transfer from the source

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer from the source is nonsecure.

Bit 14 **SAP**: source allocated port

This bit is used to allocate the master port for the source transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: source byte exchange within the unaligned half-word of each source word

If the source data width is shorter than a word, this bit is ignored.

If the source data width is a word:

0:no byte-based exchange within the unaligned half-word of each source word

1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: padding/alignment mode

If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.

Condition: PAM_1

00: source data is transferred as right aligned, padded with 0s up to the destination data width

01: source data is transferred as right aligned, sign extended up to the destination data width

10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer

Condition: PAM_2

00: source data is transferred as right aligned, left-truncated down to the destination data width

01: source data is transferred as left-aligned, right-truncated down to the destination data width

10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: source incrementing burst

0: fixed burst

1: contiguously incremented burst

The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a burst in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

17.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x. This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]					
rw	rw					rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half-transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

01: channel x (x = 0 to 3), same as 00, channel x (x = 4 to 5), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half-transfer event is generated at the half of the LLI data transfer The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x = 4 to 5), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half-transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half-transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 17.3.7*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x (x = 4 to 5): same, each block also if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x (x = 0 to 3): same as 00; channel x (x = 4 to 5): at 2D/repeated block level: the first burst read of the 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. LLI data transfer is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger:

– If the peripheral is the source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is the destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and one trigger hit.

As shown in *Figure 132: Trigger hit, memorization, and overrun waveform*, the GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10). The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled. After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 17.3.2 for the list of the implemented channels with this feature).

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see *Section 17.3.6* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 17.3.4*).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[7:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: GPDMA hardware request selection

These bits are ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1(software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 17.3.4*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active GPDMA channels (GPDMA_CxCR.EN = 1 and GPDMA_CxTR2.SWREQ = 0 for these channels). GPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

17.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

17.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 4 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]										
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.
– If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
– if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
– if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

17.8.14 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 4 to 5). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

17.8.15 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].

– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 4 to 5), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

17.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 4 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DAO[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SAO[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: destination address offset increment

The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: source address offset increment

The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

17.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 4 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRSAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

17.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory

This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.

0: no GPDMA_CxDAR update

1: GPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory

This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.

0: no GPDMA_CxLLR update

1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure

If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.

Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register GPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

17.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 4 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0:no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register GPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

17.8.20 GPDMA register map

Table 172. GPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPDMA_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																											0	0	0	0	0	0
0x04	GPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																											0	0	0	0	0	0
0x08	GPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																											0	0	0	0	0	0

Table 172. GPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	GPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																											0	0	0	0	0	0
0x10	GPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																											0	0	0	0	0	0
0x14 - 0x4C	Reserved	Reserved																															
0x50+0x80 * x (x=0 to 5)	GPDMA_ CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x54 to 0x58+0x80*x (x=0 to 5)	Reserved	Reserved																															
0x5C+0x80 * x (x=0 to 5)	GPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x60+ 0x80 * x (x=0 to 5)	GPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]								Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	DRAININGF	IDLEF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0							0	1
0x64+ 0x80 * x (x=0 to 5)	GPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	DRAINFIFO	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0					0	0	0	0
0x68 to 0x8C+0x80 * x (x=0 to 5)	Reserved	Reserved																															
0x90+0x80 * x (x=0 to 5)	GPDMA_CxTR1	DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2 [1:0]		SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2 [1:0]	
	Reset value	0	0			0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0
0x94+0x80 * x (x=0 to 5)	GPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]						TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
	Reset value	0	0					0	0			0	0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0
0x98+0x80 * x (x=0 to 5)	GPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98+0x80 * x (x=6 to 5)	GPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C+0x80 * x (x=0 to 5)	GPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0+0x80 * x (x=0 to 5)	GPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4+0x80 * x (x=4 to 5)	GPDMA_CxTR3	Res.	Res.	Res.	DAO[12:0]													Res.	Res.	Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8+0x80 * x (x=4 to 5)	GPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 172. GPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xCC+0x80 * x (x=0 to 5)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xCC+0x80 * x (x=4 to 5)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

18 Nested vectored interrupt controller (NVIC)

18.1 NVIC main features

- 148 maskable interrupt channels (not including the 16 Cortex-M33 with FPU interrupt lines)
- 16 programmable priority levels (four bits of interrupt priority used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, enabling low-latency interrupt processing and efficient processing of late-arriving interrupts.

The NVIC registers are banked across secure and nonsecure states.

All interrupts. Including the core exceptions, are managed by the NVIC.

18.2 SysTick calibration value register

The Cortex-M33 with TrustZone mainline security extension embeds two SysTick timers.

When TrustZone is activated, the following SysTick timers are available:

- SysTick, secure instance
- SysTick, nonsecure instance

When TrustZone is disabled, only one SysTick timer is available.

The SysTick timer calibration value (STCALIB) is 0x3E8. It gives a reference time base of 1 ms based on a SysTick clock frequency of 1 MHz. To match the 1 ms time base for an application running at a given frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- When SysTick clock source is CPU clock HCLK reload value = (HCLK x STCALIB) - 1
- When SysTick clock source is external clock (HCLK/8) reload value = ((HCLK / 8) x STCALIB) - 1

The HCLK refers to the AHB frequency value in MHz.

Example: SysTick clock source is CPU clock HCLK of 100 MHz, to match a time base of 1 ms: SysTick reload value = (100 x STCALIB) - 1 = 0x1869F.

18.3 Interrupt and exception vectors

For the actual availability of some peripherals, refer to the datasheet. If not present, consider them as reserved.

Table 173. STM32G56x/55x/54x/53x vector [1]table[2]

IRQ number	Security context[3]	Acronym	Description	Address
-	-	-	Reserved	0x0000 0000
-	No	Reset	Reset	0x0000 0004
-14	Yes[4]	NMI	Non maskable interrupt: HSE CSS or FLASH ECC error or RAMCFG ECC error	0x0000 0008
-13	Yes	Secure HardFault	Secure Hard fault	0x0000 000C
-13	Yes	Nonsecure HardFault	Nonsecure Hard fault. All classes of fault	0x0000 000C
-12	No	MemManage	Memory management	0x0000 0010
-11	Yes[3]	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-10	No	UsageFault	Undefined instruction or illegal state	0x0000 0018
-9	No[5]	SecureFault	SecureFault	0x0000 001C
-	-	-	Reserved	0x0000 0020
-	-	-	Reserved	0x0000 0024
-	-	-	Reserved	0x0000 0028
-5	No	SVC	System service call via SWI instruction	0x0000 002C
-4	Yes[6]	Debug Monitor	Monitor	0x0000 0030
-	-	-	Reserved	0x0000 0034
-2	No	PendSV	Pendable request for system service	0x0000 0038
-1	No	SysTick	System tick timer	0x0000 003C
0	Yes	WWDG	Window Watchdog interrupt	0x0000 0040
1	-	-	Reserved	0x0000 0044
2	Yes	IWDG	Independent watchdog interrupt	0x0000 0048
3	-	-	Reserved	0x0000 004C
4	Yes	PWR_PVD_AVD	Power Voltage Monitor/Analog Voltage Monitor	0x0000 0050
5	Yes	TAMP	Tamper global interrupts	0x0000 0054
6	Yes	GTZC1	GTZC global interrupt	0x0000 0058
7	Yes	FLASH_S	Flash secure global interrupt	0x0000 005C
8	Yes	FLASH	Flash nonsecure global interrupt	0x0000 0060
9	-	-	Reserved	0x0000 0064
10	-	-	Reserved	0x0000 0068
11	Yes	RAMCFG	RAM configuration global interrupt	0x0000 006C
12	Yes	RCC_S	RCC secure global interrupt	0x0000 0070
13	Yes	RCC	RCC nonsecure global interrupt	0x0000 0074

Table 173. STM32G56x/55x/54x/53x vector [1]table[2] (continued)

14	Yes	ICACHE	Instruction cache global interrupt	0x0000 0078
15	Yes	DCACHE	Data cache global interrupt	0x0000 007C
16	Yes	FPU	Floating point interrupt	0x0000 0080
17	Yes	RTC_S	RTC global secure interrupts	0x0000 0084
18	Yes	RTC	RTC global nonsecure interrupts	0x0000 0088
19	Yes	EXTI0	EXTI Line0 interrupt	0x0000 008C
20	Yes	EXTI1	EXTI Line1 interrupt	0x0000 0090
21	Yes	EXTI2	EXTI Line2 interrupt	0x0000 0094
22	Yes	EXTI3	EXTI Line3 interrupt	0x0000 0098
23	Yes	EXTI4	EXTI Line4 interrupt	0x0000 009C
24	Yes	EXTI5	EXTI Line5 interrupt	0x0000 00A0
25	Yes	EXTI6	EXTI Line6 interrupt	0x0000 00A4
26	Yes	EXTI7	EXTI Line7 interrupt	0x0000 00A8
27	Yes	EXTI8	EXTI Line8 interrupt	0x0000 00AC
28	Yes	EXTI9	EXTI Line9 interrupt	0x0000 00B0
29	Yes	EXTI10	EXTI Line10 interrupt	0x0000 00B4
30	Yes	EXTI11	EXTI Line11 interrupt	0x0000 00B8
31	Yes	EXTI12	EXTI Line12 interrupt	0x0000 00BC
32	Yes	EXTI13	EXTI Line13 interrupt	0x0000 00C0
33	Yes	EXTI14	EXTI Line14 interrupt	0x0000 00C4
34	Yes	EXTI15	EXTI Line15 interrupt	0x0000 00C8
35	Yes	GPDMA1_CH0	GPDMA1 channel 0 global interrupt	0x0000 00CC
36	Yes	GPDMA1_CH1	GPDMA1 channel 1 global interrupt	0x0000 00D0
37	Yes	GPDMA1_CH2	GPDMA1 channel 2 global interrupt	0x0000 00D4
38	Yes	GPDMA1_CH3	GPDMA1 channel 3 global interrupt	0x0000 00D8
39	Yes	GPDMA1_CH4	GPDMA1 channel 4 global interrupt	0x0000 00DC
40	Yes	GPDMA1_CH5	GPDMA1 channel 5 global interrupt	0x0000 00E0
41	Yes	GPDMA1_CH6	GPDMA1 channel 6 global interrupt	0x0000 00E4
42	Yes	GPDMA1_CH7	GPDMA1 channel 7 global interrupt	0x0000 00E8
43	Yes	SAES	Secure AES	0x0000 00EC
44	Yes	ADC1	ADC1 global interrupt	0x0000 00F0
45	Yes	DAC1	DAC1 global interrupt	0x0000 00F4
46	Yes	FDCAN1_IT0	FDCAN1 Interrupt 0	0x0000 00F8
47	Yes	FDCAN1_IT1	FDCAN1 Interrupt 1	0x0000 00FC
48	Yes	TIM1_BRK_TERR_IERR	TIM1 Break/TIM1 Transition error/TIM1 Index error	0x0000 0100

Table 173. STM32G56x/55x/54x/53x vector (1)table(2) (continued)

49		TIM1_UPD	TIM1 Update	0x0000 0104
50	Yes	TIM1_TRGI_ COM_DIR_IDX	TIM1 trigger and commutation/TIM1 Direction Change interrupt/TIM1 Index	0x0000 0108
51	Yes	TIM1_CC	TIM1 capture compare interrupt	0x0000 010C
52	Yes	TIM2	TIM2 global interrupt	0x0000 0110
53	Yes	TIM3	TIM3 global interrupt	0x0000 0114
54	Yes	TIM4	TIM4 global interrupt	0x0000 0118
55	Yes	TIM5	TIM5 global interrupt	0x0000 011C
56	Yes	TIM6	TIM6 global interrupt	0x0000 0120
57	Yes	TIM7	TIM7 global interrupt	0x0000 0124
58	Yes	I2C1_EV	I2C1 event interrupt	0x0000 0128
59	Yes	I2C1_ERR	I2C1 error interrupt	0x0000 012C
60	Yes	I2C2_EV	I2C2 event interrupt	0x0000 0130
61	Yes	I2C2_ERR	I2C2 error interrupt	0x0000 0134
62	Yes	SPI1	SPI1 global interrupt	0x0000 0138
63	Yes	SPI2	SPI2 global interrupt	0x0000 013C
64	Yes	SPI3	SPI3 global interrupt	0x0000 0140
65	Yes	USART1	USART1 global interrupt	0x0000 0144
66	Yes	USART2	USART2 global interrupt	0x0000 0148
67	Yes	USART3	USART3 global interrupt	0x0000 014C
68	Yes	UART4	UART4 global interrupt	0x0000 0150
69	Yes	UART5	UART5 global interrupt	0x0000 0154
70	Yes	LPUART1	LPUART1 global interrupt	0x0000 0158
71	Yes	LPTIM1	LPTIM1 global interrupt	0x0000 015C
72	Yes	TIM8_BRK_ TERR_IERR	TIM8 Break interrupt/TIM8 Transition error/TIM8 Index error	0x0000 0160
73	Yes	TIM8_UPD	TIM8 Update interrupt	0x0000 0164
74	Yes	TIM8_TRGI_ COM_DIR_IDX	TIM8 trigger and commutation interrupt/TIM8 Direction Change interrupt/TIM8 Index	0x0000 0168
75	Yes	TIM8_CC	TIM8 capture compare interrupt	0x0000 016C
76	Yes	ADC2	ADC2 global interrupt	0x0000 0170
77	Yes	ADC3	ADC3 global interrupt	0x0000 0174
78	Yes	TIM15	TIM15 global interrupt	0x0000 0178
79	Yes	TIM16	TIM16 global interrupt	0x0000 017C
80	Yes	TIM17	TIM17 global interrupt	0x0000 0180

Table 173. STM32G56x/55x/54x/53x vector [1]table[2] (continued)

81	Yes	USB_DRD_FS	USB global interrupt (dual role device, full speed)	0x0000 0184
82	Yes	CRS	Clock Recovery System global interrupt	0x0000 0188
83	Yes	UCPD1	UCPD1 global interrupt	0x0000 018C
84	Yes	-	Reserved	0x0000 0190
85	Yes	XSPI1	OCTOSPI1 global interrupt	0x0000 0194
86	Yes	TIM9	TIM9 global interrupt	0x0000 0198
87	Yes	I2C3_EV	I2C3 event interrupt	0x0000 019C
88	Yes	I2C3_ERR	I2C3 error interrupt	0x0000 01A0
89	-	-	Reserved	0x0000 01A4
90	-	-	Reserved	0x0000 01A8
91	-	-	Reserved	0x0000 01AC
92	Yes	UART6	UART6 global interrupt	0x0000 01B0
93	Yes	TIM20_BRK_ TERR_IERR	TIM20 Break/TIM1 Transition error/TIM20 Index error	0x0000 01B4
94	Yes	TIM20_UPD	TIM20 Update	0x0000 01B8
95	Yes	TIM20_TRGI_ COM_DIR_IDX	TIM20 trigger and commutation/TIM20 Direction Change interrupt/TIM20 Index	0x0000 01BC
96	Yes	TIM20_CC	TIM20 capture compare interrupt	0x0000 01C0
97	Yes	GPDMA2_CH0	GPDMA2 channel 0 global interrupt	0x0000 01C4
98	Yes	GPDMA2_CH1	GPDMA2 channel 1 global interrupt	0x0000 01C8
99	Yes	GPDMA2_CH2	GPDMA2 channel 2 global interrupt	0x0000 01CC
100	Yes	GPDMA2_CH3	GPDMA2 channe3 global interrupt	0x0000 01D0
101	Yes	GPDMA2_CH4	GPDMA2 channel 4 global interrupt	0x0000 01D4
102	Yes	GPDMA2_CH5	GPDMA2 channel 5 global interrupt	0x0000 01D8
103	Yes	GPDMA2_CH6	GPDMA2 channel 6 global interrupt	0x0000 01DC
104	Yes	GPDMA2_CH7	GPDMA2 channel 7 global interrupt	0x0000 01E0
105	-	-	Reserved	0x0000 01E4
106	-	-	Reserved	0x0000 01E8
107	Yes	FDCAN3_IT0	FDCAN3 Interrupt 0	0x0000 01EC
108	Yes	FDCAN3_IT1	FDCAN3 Interrupt 1	0x0000 01F0
109	-	-	Reserved	0x0000 01F4
110	-	-	Reserved	0x0000 01F8
111	Yes	FMAC2	FMAC2 interrupt	0x0000 01FC
112	Yes	FDCAN2_IT0	FDCAN2 Interrupt 0	0x0000 0200
113	Yes	FDCAN2_IT1	FDCAN2 Interrupt 1	0x0000 0204
114	Yes	CORDIC1	CORDIC1 interrupt	0x0000 0208

Table 173. STM32G56x/55x/54x/53x vector [1]table[2] (continued)

115	Yes	FMAC1	FMAC1 interrupt	0x0000 020C
116	Yes	DTS	DTS interrupt OR DTS AIT through EXTI line	0x0000 0210
117	Yes	RNG	RNG global interrupt	0x0000 0214
118	Yes	OTFDEC1	OTFDEC1 secure global interrupt	0x0000 0218
119	Yes	AES	AES global interrupt	0x0000 021C
120	Yes	HASH	HASH interrupt	0x0000 0220
121	Yes	PKA	PKA global interrupt	0x0000 0224
122	Yes	TIM10	TIM10 global interrupt	0x0000 0228
123	Yes	TIM12	TIM12 global interrupt	0x0000 022C
124	Yes	TIM13	TIM13 global interrupt	0x0000 0230
125	Yes	TIM14	TIM14 global interrupt	0x0000 0234
126	Yes	I3C1_EV	I3C1 event interrupt	0x0000 0238
127	Yes	I3C1_ERR	I3C1 error interrupt	0x0000 023C
128	Yes	I2C4_EV	I2C4 event interrupt	0x0000 0240
129	Yes	I2C4_ERR	I2C4 error interrupt	0x0000 0244
130	-	-	Reserved	0x0000 0248
131	-	-	Reserved	0x0000 024C
132	-	-	Reserved	0x0000 0250
133	-	-	Reserved	0x0000 0254
134	-	-	Reserved	0x0000 0258
135	-	-	Reserved	0x0000 025C
136	Yes	POSENC1	POSENC1 global interrupt	0x0000 0260
137	Yes	POSENC2	POSENC2 global interrupt	0x0000 0264
138	Yes	HRTIM1_M	HRtimer1 master interrupt	0x0000 0268
139	Yes	HRTIM1_TIMA	HRtimer1 slave1 interrupt	0x0000 026C
140	Yes	HRTIM1_TIMB	HRtimer1 slave2 interrupt	0x0000 0270
141	Yes	HRTIM1_TIMC	HRtimer1 slave3 interrupt	0x0000 0274
142	Yes	HRTIM1_TIMD	HRtimer1 slave4 interrupt	0x0000 0278
143	Yes	HRTIM1_TIME	HRtimer1 slave5 interrupt	0x0000 027C
144	Yes	HRTIM1_TIMF	HRtimer1 slave6 interrupt	0x0000 0280
145	Yes	HRTIM1_FLT	HRtimer1 error interrupt	0x0000 0284
146	Yes	PLAY1_S	PLA global secure interrupt OR play1_out14 through EXTI line	0x0000 0288
147	Yes	PLAY1	PLA global nonsecure interrupt OR play1_out14 through EXTI line	0x0000 028C
148	Yes	COMP1	COMP1 global interrupt (EXTI)	0x0000 0290

Table 173. STM32G56x/55x/54x/53x vector [1]table[2] (continued)

149	Yes	COMP2	COMP2 global interrupt (EXTI)	0x0000 0294
150	Yes	COMP3	COMP3 global interrupt (EXTI)	0x0000 0298
151	Yes	COMP4	COMP4 global interrupt (EXTI)	0x0000 029C
152	Yes	COMP5	COMP5 global interrupt (EXTI)	0x0000 02A0
153	Yes	COMP6	COMP6 global interrupt (EXTI)	0x0000 02A4
154	Yes	COMP7	COMP7 global interrupt (EXTI)	0x0000 02A8
155	Yes	COMP8	COMP8 global interrupt (EXTI)	0x0000 02AC
156	-	-	Reserved	0x0000 02B0
157	-	-	Reserved	0x0000 02B4
158	-	-	Reserved	0x0000 02B8
159	-	-	Reserved	0x0000 02BC
160	-	-	Reserved	0x0000 02C0
161	-	-	Reserved	0x0000 02C4
162	-	-	Reserved	0x0000 02C8
163	-	-	Reserved	0x0000 02CC
164	Yes	DAC2	DAC2 global interrupt	0x0000 02D0
165	Yes	-	Reserved	0x0000 02D4
166	Yes	-	Reserved	0x0000 02D8
167	Yes	DAC5	DAC5 global interrupt	0x0000 02DC
168	Yes	DAC6	DAC6 global interrupt	0x0000 02E0
169	Yes	DAC7	DAC7 global interrupt	0x0000 02E4
170	Yes	DAC8	DAC8 global interrupt	0x0000 02E8
171	Yes	DAC9	DAC9 global interrupt	0x0000 02EC
172	Yes	DAC10	DAC10 global interrupt	0x0000 02F0
173	-	-	Reserved	0x0000 02F4
174	-	-	Reserved	0x0000 02F8
175	-	-	Reserved	0x0000 02FC
176	-	-	Reserved	0x0000 0300
177	-	-	Reserved	0x0000 0304
178	-	-	Reserved	0x0000 0308
179	-	-	Reserved	0x0000 030C
180	-	-	Reserved	0x0000 0310
181	-	-	Reserved	0x0000 0314
182	-	-	Reserved	0x0000 0318
183	-	-	Reserved	0x0000 031C
184	-	-	Reserved	0x0000 0320

Table 173. STM32G56x/55x/54x/53x vector (1)table(2) (continued)

185	-	-	Reserved	0x0000 0324
186	-	-	Reserved	0x0000 0328
187	Yes	TIM11	TIM11 global interrupt	0x0000 032C
188	Yes	TIM26	TIM26 global interrupt	0x0000 0330
189	Yes	TIM27	TIM27 global interrupt	0x0000 0334

1. When different signals are connected to the same NVIC interrupt line, they are OR-ed.
2. Refer to the Programming Manual of Cortex-M33 for more information about priority.
3. Security attribute configuration can be configured as secure or nonsecure.
4. NMI and BusFault can be configured to target secure or nonsecure state using BFHFNMINS, bit13 in Cortex-M33 AIRCR register.

   For more information, refer to Arm® Technical Reference Manuals.
5. SecureFault always targets secure state.
6. DebugMonitor exception can target secure or nonsecure state depending on debug authentication signals status DBGEN/NIDEN/SPIDEN/SPNIDEN. For more information, refer to Arm® Technical Reference Manuals.

19 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wake-up through configurable and direct event inputs. It provides wake-up requests to the power control and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU, an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wake-up requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can be used also in Run modes.

The EXTI also includes the EXTI mux IO port selection.

19.1 EXTI main features

The EXTI main features are the following:

- 59 input events supported.
- All event inputs allow the possibility to wake up the system.
- Events that do not have an associated wake-up flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI.
- Events can be used to generate a CPU wake-up event.

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse), with the following features
  - Selectable active trigger edge
  - Interrupt pending status register bits independent for the rising and falling edge
  - Individual interrupt and event generation mask, used for conditioning the CPU wake-up, interrupt and event generation
  - Software trigger possibility
  - Secure events: The access to control and configuration bits of secure input events can be made secure and or privilege.
  - EXTI IO port selection
- Direct events (interrupt and wake-up sources from peripherals having an associated flag which requires to be cleared in the peripheral), with the following features:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
  - Individual interrupt generation mask, used to condition the CPU wake-up generation
  - No software trigger possibility

19.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block and EXTI mux as shown in *Figure 135*.

The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wake-up, interrupt and event outputs, and their masking.

The EXTI mux provides the IO port selection on to the EXTI event signal.

Figure 135. EXTI block diagram

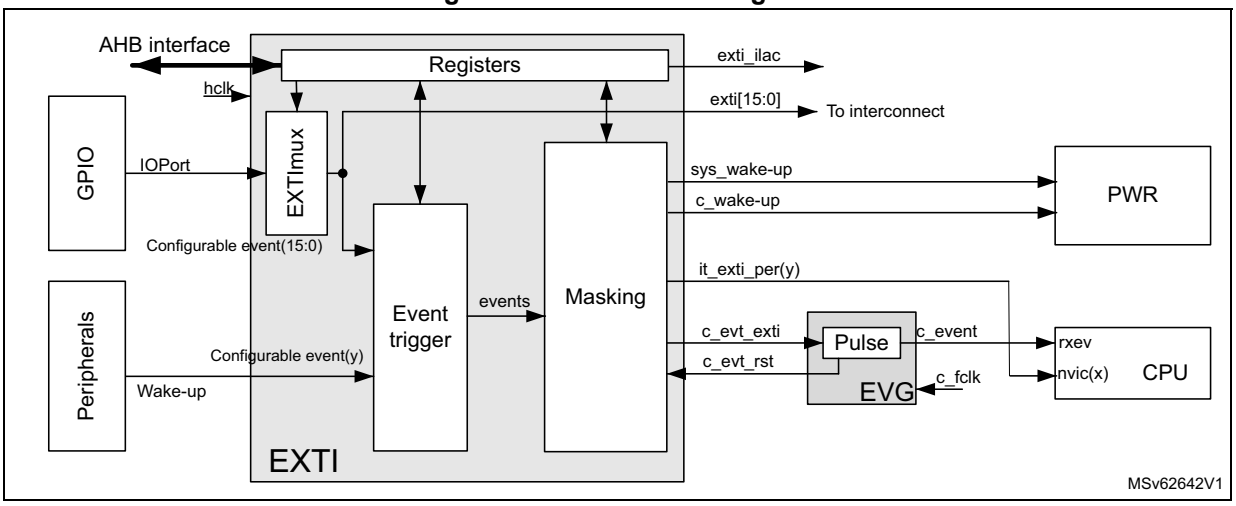


Table 174. EXTI signals

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to enable security, the AHB interface supports secure accesses.
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wake-up events from peripherals without an associated interrupt and flag
Direct event(x)	I	Synchronous and asynchronous wake-up events from peripherals with an associated interrupt and flag
exti_ilac	O	Illegal access event
IOPort(n)	I	GPIOs block IO ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other peripherals
it_exti_per (y)	O	Interrupts to the CPU associated with configurable event (y)
c_evt_exti	O	High-level sensitive event output for CPU, synchronous to hclk
c_evt_rst	I	Asynchronous reset input to clear c_evt_exti
sys_wakeup	O	Asynchronous system wake-up request to PWR for ck_sys and hclk
c_wakeup	O	Wake-up request to PWR for CPU, synchronous to hclk

Table 175. EVG signals

Pin name	I/O	Description
c_fclk	I	CPU free running clock
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_rst	O	Event reset signal, synchronous to CPU clock

19.2.1 EXTI connections between peripherals and CPU

Some peripherals able to generate wake-up or interrupt events when the system is in Stop mode are connected to the EXTI.

- Peripheral wake-up signals that generate a pulse or do not have an interrupt status bits in the peripheral are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit to be cleared. It is the EXTI interrupt, associated with the status bit, that interrupts the CPU.
- Peripheral interrupt and wake-up signals with a status bit in the peripheral to be cleared are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wake-up is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.
- All GPIO ports input to the EXTI multiplexer allow the selection of a port pin to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wake-up signals are connected to the PWR and are used to wake up the system and the CPU subsystem bus clocks.

19.2.2 EXTI interrupt/event mapping

The EXTI lines are connected as shown in the table below.

Table 176. EXTI line connections

EXTI line	Line source	Line type
0-15	GPIO	Configurable
16	PWR_PVD	Configurable
17	RTC nonsecure	Direct
18	RTC secure	Direct
19	TAMP	Direct
20	IWDG early wake-up	Direct
21	I2C1 wake-up	Direct
22	I2C2 wake-up	Direct
23	I2C3 wake-up(1)	Direct
24	I3C1 wake-up	Direct

Table 176. EXTI line connections (continued)

EXTI line	Line source	Line type
25	USART1 wake-up	Direct
26	USART2 wake-up	Direct
27	USART3 wake-up(1)	Direct
28	UART4 wake-up	Direct
29	UART5 wake-up	Direct
30	UART6 wake-up	Direct
31	Reserved	-
32	Reserved	-
33	COMP1	Configurable
34	COMP2	Configurable
35	COMP3	Configurable
36	COMP4	Configurable
37	LPUART1 wake-up	Direct
38	LPTIM1 wake-up	Direct
39	Reserved	-
40	SPI1 wake-up	Direct
41	SPI2 wake-up	Direct
42	SPI3 wake-up(1)	Direct
43	Reserved	-
44	Reserved	-
45	Reserved	-
46	Reserved	-
47	USB wake-up(1)	Direct
48	UCPD1 wake-up(1)	Direct
49	Reserved	-
50	DTS wake-up	Configurable
51	play1_out14	Configurable
52	I2C4 wake-up(1)	Direct
53	PWR_AVD	Configurable
54	COMP5	Configurable
55	COMP6	Configurable
56	COMP7	Configurable
57	COMP8	Configurable
58	LSECSSD	Direct

1. EXTI lines 23, 27, 42, 47, 48, 51, 52 (and corresponding control bits in registers) are available only on STM32G55x/56x devices.

19.3 EXTI functional description

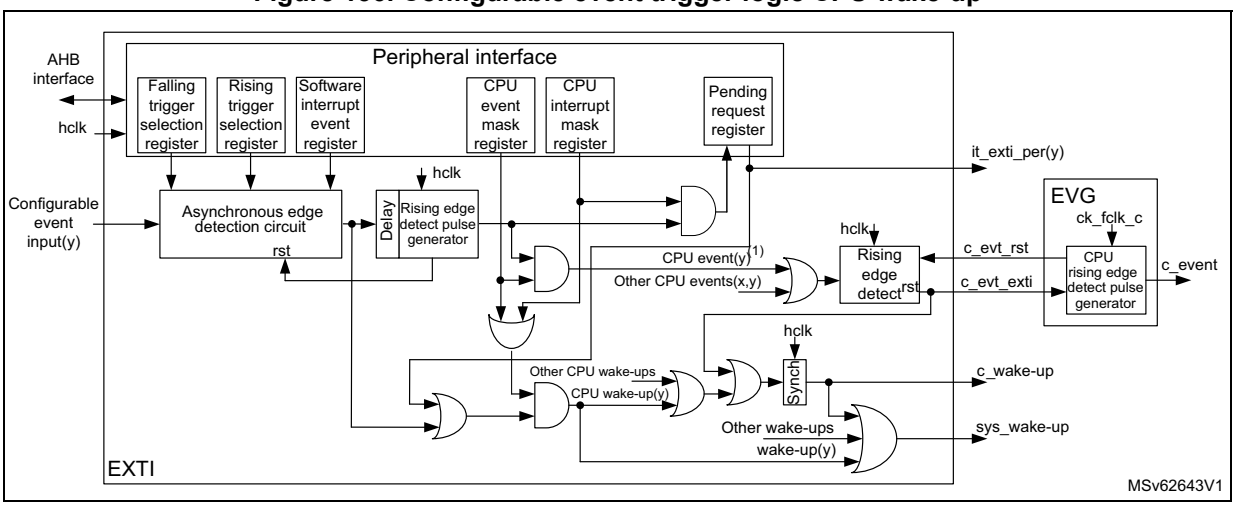
The events features are controlled from register bits as follows:

- Active trigger edge enable
  - by rising edge selection in *EXTI rising trigger selection register (EXTI_RTSR1)* and *EXTI rising trigger selection register 2 (EXTI_RTSR2)*
  - by falling edge selection in *EXTI falling trigger selection register (EXTI_FTSR1)* and *EXTI falling trigger selection register 2 (EXTI_FTSR2)*
- Software trigger in the *EXTI software interrupt event register (EXTI_SWIER1)* and *EXTI software interrupt event register 2 (EXTI_SWIER2)*
- Interrupt pending flag in
  - *EXTI rising edge pending register (EXTI_RPR1)* and *EXTI rising edge pending register 2 (EXTI_RPR2)*
  - *EXTI falling edge pending register (EXTI_FPR1)* and *EXTI falling edge pending register 2 (EXTI_FPR2)*
- CPU wake-up and interrupt enable in *EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)* and *EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2)*
- CPU wake-up and event enable in *EXTI CPU wake-up with event mask register (EXTI_EMR1)* and *EXTI CPU wake-up with event mask register 2 (EXTI_EMR2)*

19.3.1 EXTI configurable event input wake-up

The figure below details the logic associated with configurable event inputs that wake up the CPU subsystem bus clocks, and generate an EXTI pending flag and interrupt to the CPU, and/or a CPU wake-up event.

Figure 136. Configurable event trigger logic CPU wake-up



1. Only for the input events that support CPU rxev generation c_event.

The software interrupt event register allows configurable events to be triggered by software, writing the corresponding register bit, whatever the edge selection setting.

The configurable event active trigger edge (or both edges) is selected and enabled in the rising/falling edge selection registers.

The CPU has its dedicated wake-up (interrupt) mask register and a dedicated event mask registers. When the event is enabled, it is generated to the CPU. All events for the CPU are ORed together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts must be acknowledged by software in the EXTI_RPR and/or EXTI_FPR registers.

When a CPU wake-up (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

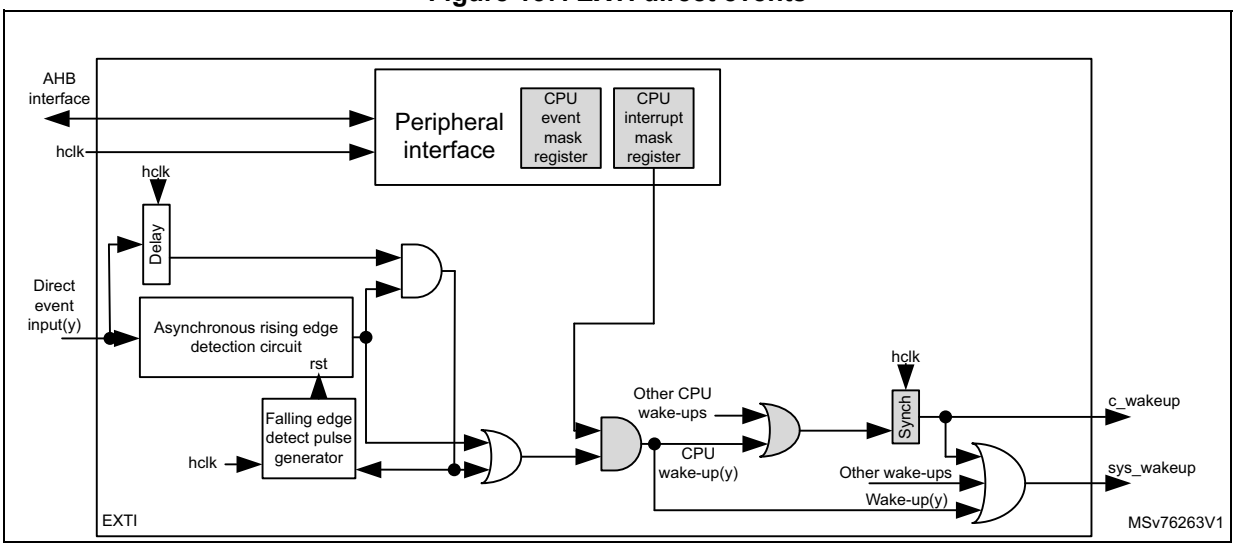
Note: A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is not able to enter low-power modes as long as an interrupt pending request is active.

19.3.2 EXTI direct event input wake-up

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU subsystem clocks. The peripheral synchronous interrupt, associated with the direct wake-up event, wakes up the CPU.

The EXTI direct event is able to wake up the CPU.

Figure 137. EXTI direct events

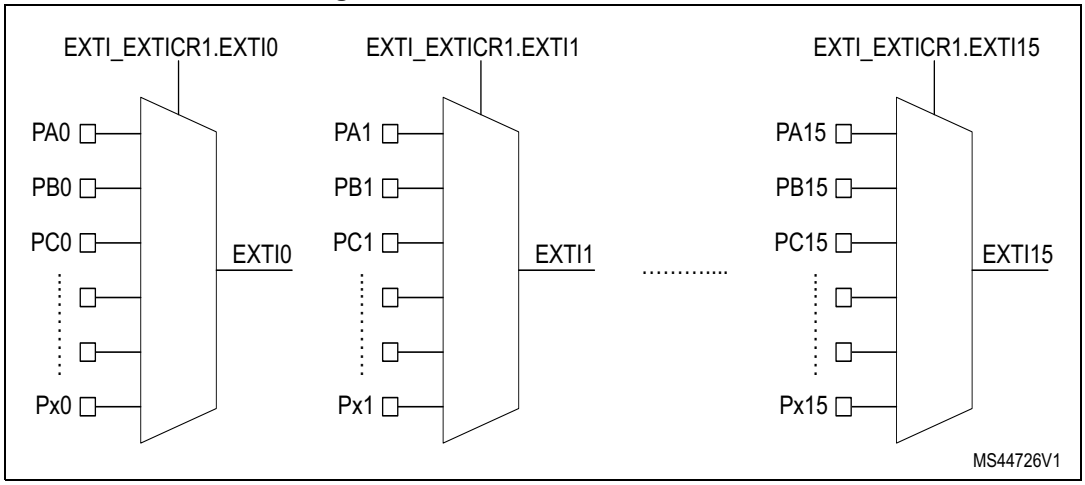


19.3.3 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wake-up. GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The

selection of GPIO port as EXTI mux output is controlled in the *EXTI external interrupt selection register (EXTI_EXTICR1)*.

Figure 138. EXTI mux GPIO selection



The EXTI mux outputs are available as output signals from the EXTI to trigger other peripherals, whatever the masking in EXTI_IMR and EXTI_EMR registers.

19.4 EXTI functional behavior

The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the CPU wake-up generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 177. Masking functionality

CPU interrupt enable (in EXTI_IMR.IMn)	CPU event enable (in EXTI_EMR.EMn)	Configurable event inputs (in EXTI_RPR.RPIFn and EXTI_FPR.FPIFn)	Exti(n) interrupt(1)	CPU event	CPU wake-up
0	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes(2)
	1	Status latched	Yes	Yes	Yes

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU NVIC.
2. Only if CPU interrupt is enabled in EXTI_IMR.IMn.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending bits EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set: the CPU subsystem is woken up and the CPU interrupt signal is activated. The EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn pending bits must be cleared by software writing them to 1. This action clears the CPU interrupt.

For the configurable event inputs, an event request can be generated by software when writing a 1 in the software interrupt/event register EXTI_SWIER, allowing the generation of a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, whatever the setting in EXTI_RTSR.

19.5 EXTI event protection

The EXTI is able to protect event register bits from being modified by nonsecure and unprivileged accesses. The protection is individually activated per input event via the register bits in EXTI_SECCFGR and EXTI_PRIVCFGR. At EXTI level, the protection consists in preventing the following unauthorized write access:

- Change the settings of the secure and/or privileged configurable events.
- Change the masking of the secure and/or privileged input events.
- Clear pending status of the secure and/or privileged input events.

Table 178. Register protection overview

Register name	Access type	Protection(1)(2)
EXTI_RTSR	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_FTSR	RW	
EXTI_SWIER	RW	
EXTI_RPR	RW	
EXTI_FPR	RW	
EXTI_SECCFGR	RW	Always secure. Privilege can be bit-wise enabled in EXTI_PRIVCFGR.
EXTI_PRIVCFGR	RW	Always privilege. Security can be bit-wise enabled in EXTI_SECCFGR.
EXTI_EXTICRn	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_LOCKR	RW	Always secure
EXTI_IM	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_EMR	RW	

1. Security is enabled with the individual input event (EXTI_SECCFGR register).
2. Privilege is enabled with the individual Input event (EXTI_PRIVCFGR register).

19.5.1 EXTI security protection

When security is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a secure access. A nonsecure write access is discarded and a read returns 0.

When input events are nonsecure, the security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and nonsecure access.

The security configuration in registers EXTI_SECCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.

19.5.2 EXTI privilege protection

When privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privileged access. An unprivileged write access is discarded and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privileged access and unprivileged access.

The privileged configuration in registers EXTI_PRIVCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.

19.6 EXTI registers

Table 179. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [58:32] configuration
0x060 - 0x06C	EXTI IO port mux selection
0x070	EXTI protection lock configuration
0x080 - 0x0BC	CPU input event configuration

All registers can be accessed with word (32-bit), half-word (16-bit), and byte (8-bit) access.

19.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RTx:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, RTx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, RTx can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

19.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FTx:** Falling trigger event configuration bit of configurable event input x [(1)] (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Nonsecure write to this FTx is discarded, nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.

0: Falling trigger disabled (for event and Interrupt) for input line

1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

19.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **SWIx:** Software interrupt on event x (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Nonsecure write to this SWI x is discarded, nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

19.6.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF16
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RPIFx:** configurable event inputs x rising edge pending bit (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access. Nonsecure write to this RPIFx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

19.6.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FPIFx:** configurable event inputs x falling edge pending bit (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Nonsecure write to this FPIFx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

19.6.6 EXTI security configuration register (EXTI_SECCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides write access security, a nonsecure write access is ignored and causes the generation of an illegal access event. A nonsecure read returns the register data.

Contains only register bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **SECx:** Security enable on event input x (x = 30 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: Event security disabled (nonsecure)
1: Event security enabled (secure)

19.6.7 EXTI privilege configuration register (EXTI_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for privilege capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **PRIVx:** Security enable on event input x (x = 30 to 0)
When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and nonsecure access.
When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. nonsecure write to this PRIVx is discarded.
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

19.6.8 EXTI rising trigger selection register 2 (EXTI_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RT57	RT56	RT55	RT54	RT53	Res.	RT51	RT50	Res.	Res.
						rw	rw	rw	rw	rw		rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT36	RT35	RT34	RT33	Res.
											rw	rw	rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:21 **RTx:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 57 to 53)
When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, RTx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RTx can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **RTx:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 51 to 50)
When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, RTx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RTx can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:1 **RTx:** Rising trigger event configuration bit of configurable event input x[1] (x = 36 to 33)
When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, RTx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RTx can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

19.6.9 EXTI falling trigger selection register 2 (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FT57	FT56	FT55	FT54	FT53	Res.	FT51	FT50	Res.	Res.
						rw	rw	rw	rw	rw		rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT36	FT35	FT34	FT33	Res.
											rw	rw	rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:21 **FTx:** Falling trigger event configuration bit of configurable event input x[1] (x = 57 to 53)
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Nonsecure write to this FTx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **FTx:** Falling trigger event configuration bit of configurable event input x[1] (x = 51 to 50)
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Nonsecure write to this FTx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:1 **FTx:** Falling trigger event configuration bit of configurable event input x[1] (x = 36 to 33)
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Nonsecure write to this FTx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

19.6.10 EXTI software interrupt event register 2 (EXTI_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SWI57	SWI56	SWI55	SWI54	SWI53	Res.	SWI51	SWI50	Res.	Res.
						rw	rw	rw	rw	rw		rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI36	SWI35	SWI34	SWI33	Res.
											rw	rw	rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:21 **SWIx:** Software interrupt on event x (x = 57 to 53)
When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Nonsecure write to this SWI x is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **SWIx:** Software interrupt on event x (x = 51 to 50)
When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Nonsecure write to this SWI x is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:1 **SWIx:** Software interrupt on event x (x = 36 to 33)
When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Nonsecure write to this SWI x is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

Bit 0 Reserved, must be kept at reset value.

19.6.11 EXTI rising edge pending register 2 (EXTI_RPR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RPIF57	RPIF56	RPIF55	RPIF54	RPIF53	Res.	RPIF51	RPIF50	Res.	Res.
						rw	rw	rw	rw	rw		rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF36	RPIF35	RPIF34	RPIF33	Res.
											rw	rw	rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:21 **RPIFx:** configurable event inputs x rising edge pending bit (x = 57 to 53)

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access. Nonsecure write to this RPIFx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **RPIFx:** configurable event inputs x rising edge pending bit (x = 51 to 50)

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access. Nonsecure write to this RPIFx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:1 **RPIFx:** configurable event inputs x rising edge pending bit (x = 36 to 33)
When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access. Nonsecure write to this RPIFx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 0 **Reserved, must be kept at reset value.**

19.6.12 EXTI falling edge pending register 2 (EXTI_FPR2)

Address offset: 0x030

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FPIF57	FPIF56	FPIF55	FPIF54	FPIF53	Res.	FPIF51	FPIF50	Res.	Res.
						rw	rw	rw	rw	rw		rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF36	FPIF35	FPIF34	FPIF33	Res.
											rw	rw	rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:21 **FPIFx:** configurable event inputs x falling edge pending bit (x = 57 to 53)
When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Nonsecure write to this FPIFx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **FPIFx:** configurable event inputs x falling edge pending bit (x = 51 to 50)
When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Nonsecure write to this FPIFx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:1 **FPIFx:** configurable event inputs x falling edge pending bit (x = 36 to 33)
When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Nonsecure write to this FPIFx is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 0 Reserved, must be kept at reset value.

19.6.13 EXTI security configuration register 2 (EXTI_SECCFGR2)

Address offset: 0x034

Reset value: 0x0000 0000

This register provides write access security, a nonsecure write access is ignored and causes the generation of an illegal access event. A nonsecure read returns the register data.

Contains only register bits for privilege capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SEC58	SEC57	SEC56	SEC55	SEC54	SEC53	SEC52	SEC51	SEC50	Res.	SEC48
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC47	Res.	Res.	Res.	Res.	SEC42	SEC41	SEC40	Res.	SEC38	SEC37	SEC36	SEC35	SEC34	SEC33	Res.
rw					rw	rw	rw		rw	rw	rw	rw	rw	rw	

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:18 **SECx:** Security enable on event input x (x = 58 to 50)
When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: Event security disabled (nonsecure)
1: Event security enabled (secure)

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **SECx:** Security enable on event input x (x = 48 to 47)
When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: Event security disabled (nonsecure)
1: Event security enabled (secure)

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **SECx:** Security enable on event input x (x = 42 to 40)
When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: Event security disabled (nonsecure)
1: Event security enabled (secure)

Bit 7 Reserved, must be kept at reset value.

Bits 6:1 **SECx:** Security enable on event input x (x = 38 to 33)
When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: Event security disabled (nonsecure)
1: Event security enabled (secure)

Bit 0 Reserved, must be kept at reset value.

19.6.14 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2)

Address offset: 0x038

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PRIV58	PRIV57	PRIV56	PRIV55	PRIV54	PRIV53	PRIV52	PRIV51	PRIV50	Res.	PRIV48
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV47	Res.	Res.	Res.	Res.	PRIV42	PRIV41	PRIV40	Res.	PRIV38	PRIV37	PRIV36	PRIV35	PRIV34	PRIV33	Res.
rw					rw	rw	rw		rw	rw	rw	rw	rw	rw	

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:18 **PRIVx:** Security enable on event input x (x = 58 to 50)
When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and nonsecure access.
When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. Nonsecure write to this PRIVx is discarded.
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **PRIVx:** Security enable on event input x (x = 48 to 47)
When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and nonsecure access.
When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. Nonsecure write to this PRIVx is discarded.
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **PRIVx:** Security enable on event input x (x = 42 to 40)
When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and nonsecure access.
When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. Nonsecure write to this PRIVx is discarded.
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bit 7 Reserved, must be kept at reset value.

Bits 6:1 **PRIVx:** Security enable on event input x (x = 38 to 33)
When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and nonsecure access.
When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. Nonsecure write to this PRIVx is discarded.
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

Bit 0 Reserved, must be kept at reset value.

19.6.15 EXTI external interrupt selection register (EXTI_EXTICR1)

Address offset: 0x060

(EXTI mux 0, 1, 2, 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection

These bits are written by software to select the source input for EXTI3 external interrupt.
When EXTI_SECCFGR1.SEC3 is disabled, EXTI3 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC3 is enabled, EXTI3 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV3 is disabled, EXTI3 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV3 is enabled, EXTI3 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA3 pin
0x01: PB3 pin
0x02: PC3 pin
0x03: PD3 pin
0x04: PE3 pin
0x05: PF3 pin
0x06: PG3 pin
Others: reserved

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.
When EXTI_SECCFGR1.SEC2 is disabled, EXTI2 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC2 is enabled, EXTI2 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV2 is disabled, EXTI2 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV2 is enabled, EXTI2 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA2 pin
0x01: PB2 pin
0x02: PC2 pin
0x03: PD2 pin
0x04: PE2 pin
0x05: PF2 pin
0x06: PG2 pin
Others: reserved

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.
When EXTI_SECCFGR1.SEC1 is disabled, EXTI1 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC1 is enabled, EXTI1 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV1 is disabled, EXTI1 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV1 is enabled, EXTI1 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA1 pin
0x01: PB1 pin
0x02: PC1 pin
0x03: PD1 pin
0x07: PH1 pin
Others: reserved

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI0 external interrupt.
When EXTI_SECCFGR1.SEC0 is disabled, EXTI0 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC0 is enabled, EXTI0 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV0 is disabled, EXTI0 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV0 is enabled, EXTI0 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA0 pin
0x01: PB0 pin
0x02: PC0 pin
0x03: PD0 pin
0x05: PF0 pin
0x07: PH0 pin
Others: reserved

19.6.16 EXTI external interrupt selection register (EXTI_EXTICR2)

Address offset: 0x064

(EXTI mux 4, 5, 6, 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection

These bits are written by software to select the source input for EXTI7 external interrupt.
When EXTI_SECCFGR1.SEC7 is disabled, EXTI7 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC7 is enabled, EXTI7 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV7 is disabled, EXTI7 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV7 is enabled, EXTI7 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA7 pin
0x01: PB7 pin
0x02: PC7 pin
0x03: PD7 pin
0x04: PE7 pin
0x05: PF7 pin
Others: reserved

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection

These bits are written by software to select the source input for EXTI6 external interrupt.
When EXTI_SECCFGR1.SEC6 is disabled, EXTI6 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC6 is enabled, EXTI6 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV6 is disabled, EXTI6 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV6 is enabled, EXTI6 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA6 pin
0x01: PB6 pin
0x02: PC6 pin
0x03: PD6 pin
0x04: PE6 pin
Others: reserved

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection

These bits are written by software to select the source input for EXTI5 external interrupt.
When EXTI_SECCFGR1.SEC5 is disabled, EXTI5 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC5 is enabled, EXTI5 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV5 is disabled, EXTI5 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV5 is enabled, EXTI5 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA5 pin
0x01: PB5 pin
0x02: PC5 pin
0x03: PD5 pin
0x04: PE5 pin
0x06: PG5 pin
0x07: PH5 pin
Others: reserved

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection

These bits are written by software to select the source input for EXTI4 external interrupt.
When EXTI_SECCFGR1.SEC4 is disabled, EXTI4 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC4 is enabled, EXTI4 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV4 is disabled, EXTI4 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV4 is enabled, EXTI4 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA4 pin
0x01: PB4 pin
0x02: PC4 pin
0x03: PD4 pin
0x04: PE4 pin
0x05: PF4 pin
0x06: PG4 pin
Others: reserved

19.6.17 EXTI external interrupt selection register (EXTI_EXTICR3)

Address offset: 0x068

(EXTI mux 8, 9, 10, 11)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection

These bits are written by software to select the source input for EXTI11 external interrupt.
When EXTI_SECCFGR1.SEC11 is disabled, EXTI11 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC11 is enabled, EXTI11 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV11 is disabled, EXTI11 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV11 is enabled, EXTI11 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA11 pin
0x01: PB11 pin
0x02: PC11 pin
0x03: PD11 pin
0x04: PE11 pin
0x05: PF11 pin
0x06: PG11 pin
Others: reserved

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection

These bits are written by software to select the source input for EXTI10 external interrupt.
When EXTI_SECCFGR1.SEC10 is disabled, EXTI10 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC10 is enabled, EXTI10 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV10 is disabled, EXTI10 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV10 is enabled, EXTI10 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA10 pin
0x01: PB10 pin
0x02: PC10 pin
0x03: PD10 pin
0x04: PE10 pin
0x05: PF10 pin
0x06: PG10 pin
Others: reserved

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI9 external interrupt.
When EXTI_SECCFGR1.SEC9 is disabled, EXTI9 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC9 is enabled, EXTI9 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV9 is disabled, EXTI9 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV9 is enabled, EXTI9 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA9 pin
0x01: PB9 pin
0x02: PC9 pin
0x03: PD9 pin
0x04: PE9 pin
0x05: PF9 pin
0x06: PG9 pin
Others: reserved

Bits 7:0 **EXTI8[7:0]:** EXTI8 GPIO port selection

These bits are written by software to select the source input for EXTI8 external interrupt.
When EXTI_SECCFGR1.SEC8 is disabled, EXTI8 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC8 is enabled, EXTI8 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV8 is disabled, EXTI8 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV8 is enabled, EXTI8 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA8 pin
0x02: PC8 pin
0x03: PD8 pin
0x04: PE8 pin
0x05: PF8 pin
Others: reserved

19.6.18 EXTI external interrupt selection register (EXTI_EXTICR4)

Address offset: 0x060

(EXTI mux 12, 13, 14, 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection

These bits are written by software to select the source input for EXTI15 external interrupt.
When EXTI_SECCFGR1.SEC15 is disabled, EXTI15 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC15 is enabled, EXTI15 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV15 is disabled, EXTI15 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV15 is enabled, EXTI15 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA15 pin
0x01: PB15 pin
0x02: PC15 pin
0x03: PD15 pin
0x04: PE15 pin
0x05: PF15 pin
Others: reserved

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection

These bits are written by software to select the source input for EXTI14 external interrupt.
When EXTI_SECCFGR1.SEC14 is disabled, EXTI14 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC14 is enabled, EXTI14 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV14 is disabled, EXTI14 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV14 is enabled, EXTI14 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA14 pin
0x01: PB14 pin
0x02: PC14 pin
0x03: PD14 pin
0x04: PE14 pin
0x05: PF14 pin
0x06: PG14 pin
Others: reserved

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection

These bits are written by software to select the source input for EXTI13 external interrupt.
When EXTI_SECCFGR1.SEC13 is disabled, EXTI13 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC13 is enabled, EXTI13 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV13 is disabled, EXTI13 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV13 is enabled, EXTI13 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA13 pin
0x01: PB13 pin
0x02: PC13 pin
0x03: PD13 pin
0x04: PE13 pin
0x05: PF13 pin
0x06: PG13 pin
Others: reserved

Bits 7:0 **EXTI12[7:0]:** EXTI12 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.
When EXTI_SECCFGR1.SEC12 is disabled, EXTI12 can be accessed with nonsecure and secure access.
When EXTI_SECCFGR1.SEC12 is enabled, EXTI12 can only be accessed with secure access. Nonsecure write is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR1.PRIV12 is disabled, EXTI12 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV12 is enabled, EXTI12 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA12 pin
0x01: PB12 pin
0x02: PC12 pin
0x03: PD12 pin
0x04: PE12 pin
0x05: PF12 pin
0x06: PG12 pin
Others: reserved

19.6.19 EXTI lock register (EXTI_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register provides write access security: a nonsecure write access is ignored, a read access returns zero data, and both generate an illegal access event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LOCK**: Global security and privilege configuration registers (EXTI_SECCFGR and EXTI_PRIVCFGR) lock
This bit is written once after reset.
0: Security and privilege configuration open, can be modified.
1: Security and privilege configuration locked, can no longer be modified.

19.6.20 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0x7FFE 0000

Contains register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **IMx:** CPU wake-up with interrupt mask on event input x [1] (x = 30 to 0)
When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Nonsecure write to this bit is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with interrupt request from input event x is masked.
1: Wake-up with interrupt request from input event x is unmasked.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

19.6.21 EXTI CPU wake-up with event mask register (EXTI_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **EMx:** CPU wake-up with event generation mask on event input x (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with event generation from Line x is masked.

1: Wake-up with event generation from Line x is unmasked.

19.6.22 EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2)

Address offset: 0x090

Reset value: 0x0411 8760

Contains register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	IM58	IM57	IM56	IM55	IM54	IM53	IM52	IM51	IM50	Res.	IM48
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	Res.	Res.	Res.	Res.	IM42	IM41	IM40	Res.	IM38	IM37	IM36	IM35	IM34	IM33	Res.
rw					rw	rw	rw		rw	rw	rw	rw	rw	rw	

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:18 **IMx:** CPU wake-up with interrupt mask on event input x[(1)] (x = 58 to 50)

When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Nonsecure write to this bit is discarded and nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with interrupt request from input event x is masked.

1: Wake-up with interrupt request from input event x is unmasked.

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **IMx:** CPU wake-up with interrupt mask on event input x[1] (x = 48 to 47)
When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Nonsecure write to this bit is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with interrupt request from input event x is masked.
1: Wake-up with interrupt request from input event x is unmasked.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **IMx:** CPU wake-up with interrupt mask on event input x[1] (x = 42 to 40)
When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Nonsecure write to this bit is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with interrupt request from input event x is masked.
1: Wake-up with interrupt request from input event x is unmasked.

Bit 7 Reserved, must be kept at reset value.

Bits 6:1 **IMx:** CPU wake-up with interrupt mask on event input x[1] (x = 38 to 33)
When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Nonsecure write to this bit is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with interrupt request from input event x is masked.
1: Wake-up with interrupt request from input event x is unmasked.

Bit 0 Reserved, must be kept at reset value.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

19.6.23 EXTI CPU wake-up with event mask register 2 (EXTI_EMR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	EM57	EM56	EM55	EM54	EM53	Res.	EM51	EM50	Res.	Res.
						rw	rw	rw	rw	rw		rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM36	EM35	EM34	EM33	Res.
											rw	rw	rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:21 **EMx:** CPU wake-up with event generation mask on event input x (x = 57 to 53)
When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with event generation from Line x is masked.
1: Wake-up with event generation from Line x is unmasked.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EMx:** CPU wake-up with event generation mask on event input x (x = 51 to 50)
When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with event generation from Line x is masked.
1: Wake-up with event generation from Line x is unmasked.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:1 **EMx:** CPU wake-up with event generation mask on event input x (x = 36 to 33)
When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access. Nonsecure write to this bit x is discarded and nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wake-up with event generation from Line x is masked.
1: Wake-up with event generation from Line x is unmasked.

Bit 0 Reserved, must be kept at reset value.

19.6.24 EXTI register map

Table 180. EXTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	EXTI_RPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF16	RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	EXTI_FPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16	FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	EXTI_ SECCFGR1	Res.	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	EXTI_ PRIVCFGR1	Res.	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	EXTI_RTSR2	Res.	Res.	Res.	Res.	Res.	Res.	RT57	RT56	RT55	RT54	RT53	Res.	RT51	RT50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT36	RT35	RT34	RT33	Res.
	Reset value							0	0	0	0	0		0	0														0	0	0	0	0	
0x024	EXTI_FTSR2	Res.	Res.	Res.	Res.	Res.	Res.	FT57	FT56	FT55	FT54	FT53	Res.	FT51	FT50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT37	FT35	FT34	FT33	Res.
	Reset value							0	0	0	0	0		0	0														0	0	0	0	0	
0x028	EXTI_SWIER2	Res.	Res.	Res.	Res.	Res.	Res.	SWI57	SWI56	SWI55	SWI54	SWI53	Res.	SWI51	SWI50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI36	SWI35	SWI34	SWI33	Res.
	Reset value							0	0	0	0	0		0	0														0	0	0	0	0	
0x02C	EXTI_RPR2	Res.	Res.	Res.	Res.	Res.	Res.	RPIF57	RPIF56	RPIF55	RPIF54	RPIF53	Res.	RPIF51	RPIF50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF36	RPIF35	RPIF34	RPIF33	Res.
	Reset value							0	0	0	0	0		0	0														0	0	0	0	0	
0x030	EXTI_FPR2	Res.	Res.	Res.	Res.	Res.	Res.	FPIF57	FPIF56	FPIF55	FPIF54	FPIF53	Res.	FPIF51	FPIF50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF36	FPIF35	FPIF34	FPIF33	Res.
	Reset value							0	0	0	0	0		0	0														0	0	0	0	0	
0x034	EXTI_ SECCFGR2	Res.	Res.	Res.	Res.	Res.	SEC58	SEC57	SEC56	SEC55	SEC54	SEC53	SEC52	SEC51	SEC50	Res.	SEC48	Res.	SEC47	Res.	Res.	Res.	Res.	SEC42	SEC41	SEC40	Res.	SEC38	Res.	SEC37	SEC36	SEC35	SEC34	SEC33	Res.
	Reset value						0	0	0	0	0	0	0	0	0		0	0					0	0	0		0		0	0	0	0	0	
0x038	EXTI_ PRIVCFGR2	Res.	Res.	Res.	Res.	Res.	PRIV58	PRIV57	PRIV56	PRIV55	PRIV54	PRIV53	PRIV52	PRIV51	PRIV50	Res.	PRIV48	PRIV47	Res.	Res.	Res.	Res.	PRIV42	PRIV41	PRIV40	Res.	PRIV38	PRIV37	PRIV36	PRIV35	PRIV34	PRIV33	Res.
	Reset value						0	0	0	0	0	0	0	0	0		0	0					0	0	0		0	0	0	0	0	0	
0x03C to 0x05C	Reserved	Reserved																															

Table 180. EXTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x060	EXTI_EXTICR1	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]								EXTI0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	EXTI_EXTICR2	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]								EXTI4[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	EXTI_EXTICR3	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]								EXTI8[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	EXTI_EXTICR4	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]								EXTI12[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	EXTI_LOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
	Reset value																																0
0x074 to 0x07C	Reserved	Reserved																															
0x080	EXTI_IMR1	Res.	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value		1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	EXTI_EMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	EXTI_IMR2	Res.	Res.	Res.	Res.	Res.	IM58	IM57	IM56	IM55	IM54	IM53	IM52	IM51	IM50	Res.	IM48	IM47	Res.	Res.	Res.	Res.	IM42	IM41	IM40	Res.	IM38	IM37	IM36	IM35	IM34	IM33	Res.
	Reset value						1	0	0	0	0	0	1	0	0		1	1					1	1	1		1	1	0	0	0	0	
0x094	EXTI_EMR2	Res.	Res.	Res.	Res.	Res.	Res.	EM57	EM56	EM55	EM54	EM53	Res.	EM51	EM50	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM36	EM35	EM34	EM33	Res.
	Reset value						0	0	0	0	0	0		0	0														0	0	0	0	

Refer to *Section 2.3* for the register boundary addresses.

20 Cyclic redundancy check calculation unit (CRC)

20.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

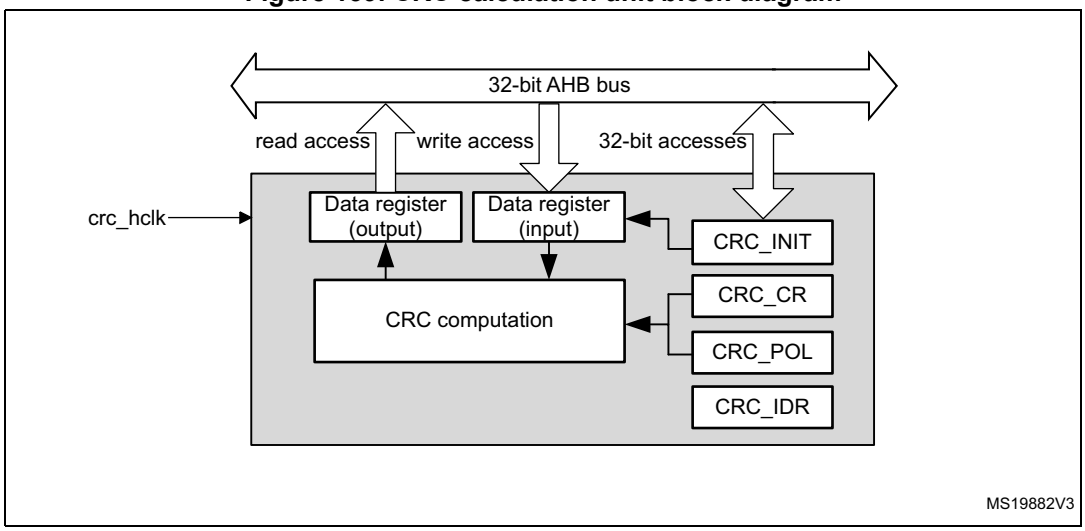
20.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility options on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

20.3 CRC functional description

20.3.1 CRC block diagram

Figure 139. CRC calculation unit block diagram



20.3.2 CRC internal signals

Table 181. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

20.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. When the RTYPE_IN bit is set in CRC_CR, non-word accesses to CRC_DR register are ignored. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on input data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register. If RTYPE_IN bit is set in this register, the input reversing operation can be performed with a byte or half-word granularity, but not with single bit granularity.

For example, with RTYPE_IN cleared, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte (REV_IN[1:0] = 01)
- 0xD458B23C with bit-reversal done by half-word (REV_IN[1:0] = 10)
- 0xB23CD458 with bit-reversal done on the full word (REV_IN[1:0] = 11)

With RTYPE_IN set, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x3C4D1A2B with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x4D3C2B1A with byte-reversal done by word (REV_IN[1:0] = 10)

The output data can also be reversed by setting the REV_OUT[1:0] bits in the CRC_CR register. If RTYPE_OUT is set in this register, the output reversing operation is performed with a byte or half-word granularity, but not with single bit granularity.

With RTYPE_OUT cleared, the operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

With RTYPE_OUT set, 0x11223344 output data are converted as:

- 0x33441122 with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x44332211 with byte-reversal done by word (REV_IN[1:0] = 10)

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC 7-, 8-, 16-, and 32-bit calculations. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7-, 8-, 16-, or 32-bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

20.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes when RTYPE bit is cleared in CRC_CR. When RTYPE is set, only word accesses are allowed. For the other registers only 32-bit accesses are allowed.

20.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

20.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

20.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
					rw	rw	rw	rw	rw	rw	rw	rw			rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RTYPE_OUT**: Reverse type output
This bit controls the reversal granularity of the output data.
0: Bit level output
1: Byte or half-word level output

Bit 9 **RTYPE_IN**: Reverse type input
This bit controls the reversal granularity of the input data.
0: Bit level input
1: Byte or half-word level input

Bits 8:7 **REV_OUT[1:0]**: Reverse output data
This bitfield controls the reversal of the bit order of the output data.
00: Bit order not affected (RTYPE_OUT = 0 or 1)
01: Bit-reversed output format (RTYPE_OUT = 0) or half-word reversal done by word (RTYPE_OUT = 1)
10: Bit order not affected (RTYPE_OUT = 0) or byte reversal done by word (RTYPE_OUT = 1)
11: Bit order not affected (RTYPE_OUT = 0 or 1)

Bits 6:5 **REV_IN[1:0]**: Reverse input data
This bitfield controls the reversal of the bit order of the input data
00: Bit order not affected (RTYPE_IN = 0 or 1)
01: Bit reversal done by byte (RTYPE_IN = 0) or half-word reversal done by word (RTYPE_IN = 1)
10: Bit reversal done by half-word (RTYPE_IN = 0) or byte reversal done by word (RTYPE_IN = 1)
11: Bit reversal done by word (RTYPE_IN = 0) or bit order is not affected (RTYPE_IN = 1)

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware.

20.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value

This register is used to write the CRC initial value.

20.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

20.4.6 CRC register map

Table 182. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																						0	0	0	0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

21 CORDIC coprocessor (CORDIC)

21.1 CORDIC introduction

The CORDIC coprocessor provides hardware acceleration of mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing, and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making possible the use of a lower operating frequency, or freeing up processor cycles to perform other tasks.

21.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atanh, phase, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

21.3 CORDIC functional description

21.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle θ are determined by rotating the unit vector [1, 0] through decreasing angles until the cumulative sum of the rotation angles equals the input angle θ. The x and y cartesian components of the rotated vector then correspond, respectively, to the cosine and sine of θ. Inversely, the angle of a vector [x, y] corresponding to arctangent (y / x), is determined by rotating [x, y] through successively decreasing angles to obtain the unit vector [1, 0]. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions (sinh, cosh, atanh), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

21.3.2 CORDIC functions

The first step when using the coprocessor is to select the required function, by programming the FUNC field of the CORDIC_CR register.

Table 183 lists the functions supported by the CORDIC coprocessor.

Table 183. CORDIC functions

Function	Primary argument (ARG1)	Secondary argument (ARG2)	Primary result (RES1)	Secondary result (RES2)
Cosine	angle θ	modulus m	$m \cdot \cos\theta$	$m \cdot \sin\theta$
Sine	angle θ	modulus m	$m \cdot \sin\theta$	$m \cdot \cos\theta$
Phase	x	y	atan2(y,x)	$\sqrt{x^2+y^2}$
Modulus	x	y	$\sqrt{x^2+y^2}$	atan2(y,x)
Arctangent	x	none	atan x	none
Hyperbolic cosine	x	none	cosh x	sinh x
Hyperbolic sine	x	none	sinh x	cosh x
Hyperbolic arctangent	x	none	atanh x	none
Natural logarithm	x	none	ln x	none
Square root	x	none	$\sqrt{x}$	none

Several functions take two input arguments (ARG1 and ARG2), and some generate two results (RES1 and RES2) simultaneously. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: sin θ also generates cos θ, and cos θ also generates sin θ. Similarly for rectangular-to-polar conversion (phase(x,y), modulus(x,y)) and for hyperbolic functions (cosh θ, sinh θ).

Note: The exponential function, exp x, can be obtained as the sum of sinh x and cosh x. Furthermore, base N logarithms, $\log_N x$, can be derived by multiplying ln x by a constant K, where K = 1/ln N.

For certain functions (atan, log, sqrt) a scaling factor (see *Section 21.3.4*) can be applied to extend the range beyond the maximum [-1, 1] supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

Cosine

Table 184. Cosine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m • cos θ	[-1, 1]

Table 184. Cosine parameters (continued)

Parameter	Description	Range
RES2	m • sin θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

Sine

Table 185. Sine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m • sin θ	[-1, 1]
RES2	m • cos θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the sine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

Phase

Table 186. Phase parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Phase angle θ in radians, divided by π	[-1, 1]

Table 186. Phase parameters (continued)

Parameter	Description	Range
RES2	Modulus m	[0, 1]
SCALE	Not applicable	0

This function calculates the phase angle in the range -π to π of a vector **v** = [x y] (also known as atan2(y,x). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the phase angle θ of the vector **v**. RES1 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

The secondary result, RES2, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES2 is saturated to 1.

Modulus

Table 187. Modulus parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Modulus m	[0, 1]
RES2	Phase angle θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the magnitude, or modulus, of a vector **v** = [x y]. It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES1 is saturated to 1.

The secondary result, RES2, is the phase angle θ of the vector **v**. RES2 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

Arctangent

Table 188. Arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-1, 1]
ARG2	Not applicable	-
RES1	2^{-n} • atan x, in radians, divided by p	[-1, 1]
RES2	Not applicable	-
SCALE	n	[0 7]

This function calculates the arctangent, or inverse tangent, of the input argument x.

The primary argument, ARG1, is the input value, x = tan θ. If |x| > 1, a scaling factor of 2^{-n} must be applied in software such that $-1 < x \cdot 2^{-n} < 1$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the scale factor n must be programmed in the SCALE parameter.

Note that the maximum input value allowed is tan θ = 128, which corresponds to an angle θ = 89.55 degrees. For |x| > 128, a software method must be used to find atan x.

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle θ = atan x. RES1 must be multiplied by $2^n \cdot \pi$ to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

Table 189. Hyperbolic cosine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \cosh x$	[0.5 0.846]
RES2	$2^{-n} \cdot \sinh x$	[-0.683 0.683]
SCALE	n	1

This function calculates the hyperbolic cosine of a hyperbolic angle x. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. Since the minimum value of cosh x is 1, which is beyond the range of the q1.31 format, a scaling factor of 2^{-n} must be applied in software. The factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic cosine, cosh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, sinh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic sine

Table 190. Hyperbolic sine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559, 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \sinh x$	[-0.683, 0.683]
RES2	$2^{-n} \cdot \cosh x$	[0.5, 0.846]
SCALE	n	1

This function calculates the hyperbolic sine of a hyperbolic angle x. It can also be used to calculate the exponential functions e^x = cosh x + sinh x, and e^{-x} = cosh x- sinh x.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. For all input values, a scaling factor of 2^{-n} must be applied in software, where n = 1. The scaled value x • 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, sinh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, cosh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic arctangent

Table 191. Hyperbolic arctangent parameters

Parameter	Description	Range
ARG1	$x \bullet 2^{-n}$	[-0.403 0.403]
ARG2	Not applicable	-
RES1	$2^{-n} \bullet$atanh x	[-0.559 0.559]
RES2	Not applicable	-
SCALE	n	1

This function calculates the hyperbolic arctangent of the input argument x.

The primary argument is the input value x. Only values of x in the -0.806 to +0.806 range are supported. The value x must be scaled by a factor 2^{-n}, where n = 1. The scaled value

x • 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, atanh x. RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

Natural logarithm

Table 192. Natural logarithm parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.054 0.875]
ARG2	Not applicable	-
RES1	$2^{-(n+1)}.\ln x$	[-0.279 0.137]
RES2	Not applicable	-
SCALE	n	[1 4]

This function calculates the natural logarithm of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.107 to 9.35 are supported. The value x must be scaled by a factor 2^{-n}, such that $x < 2^{-n} < 1\text{-}2^{-n}$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 193 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 193. Natural log scaling factors and corresponding ranges

n	x range	ARG1 range
1	$0.107 \le x < 1$	$0.0535 \le ARG1 < 0.5$
2	$1 \le x < 3$	$0.25 \le ARG1 < 0.75$
3	$3 \le x < 7$	$0.375 \le ARG1 < 0.875$
4	$7 \le x \le 9.35$	$0.4375 \le ARG1 < 0.584$

The secondary argument is not used.

The primary result is the natural logarithm, ln x. RES1 must be multiplied by $2^{(n+1)}$ to obtain the correct value.

The secondary result is not used.

Square root

Table 194. Square root parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.027 0.875]
ARG2	Not applicable	-
RES1	$2^{-n}\sqrt{x}$	[0.04 1]
RES2	Not applicable	-
SCALE	n	[0 2]

This function calculates the square root of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.027 to 2.34 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < (1 - 2^{(-n-2)})$.

The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 195 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 195. Square root scaling factors and corresponding ranges

n	x range	ARG1 range
0	$0.027 \leq x < 0.75$	$0.027 \leq ARG1 < 0.75$
1	$0.75 \leq x < 1.75$	$0.375 \leq ARG1 < 0.875$
2	$1.75 \leq x \leq 2.341$	$0.4375 \leq ARG1 \leq 0.585$

The secondary argument is not used.

The primary result is the square root of x. RES1 must be multiplied by 2^n to obtain the correct value.

The secondary result is not used.

21.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore -1 (0x80000000) to $1 - 2^{-31}$ (0x7FFFFFFF).

In q1.15 format, the numeric range is 1 (0x8000) to $1 - 2^{-15}$ (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

21.3.4 Scaling factor

Several of the functions listed in *Section 21.3.2* specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output, or internal registers. If the scaling factor is

required, it must be calculated by software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

Note: The scaling factor entails a loss of precision due to truncation of the scaled value.

21.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see *Figure 140*.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see *Figure 141*). The square root function converges at roughly twice the speed of the hyperbolic functions (see *Figure 142*).

Figure 140. CORDIC convergence for trigonometric functions

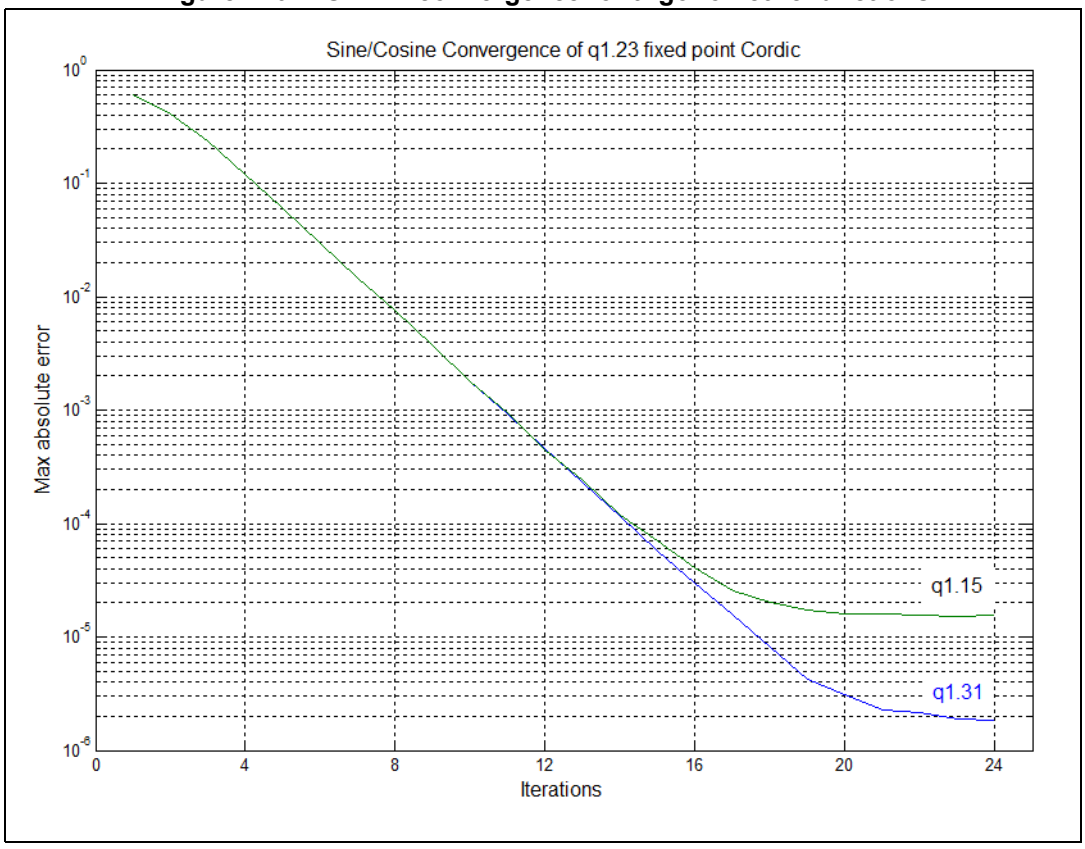


Figure 141. CORDIC convergence for hyperbolic functions

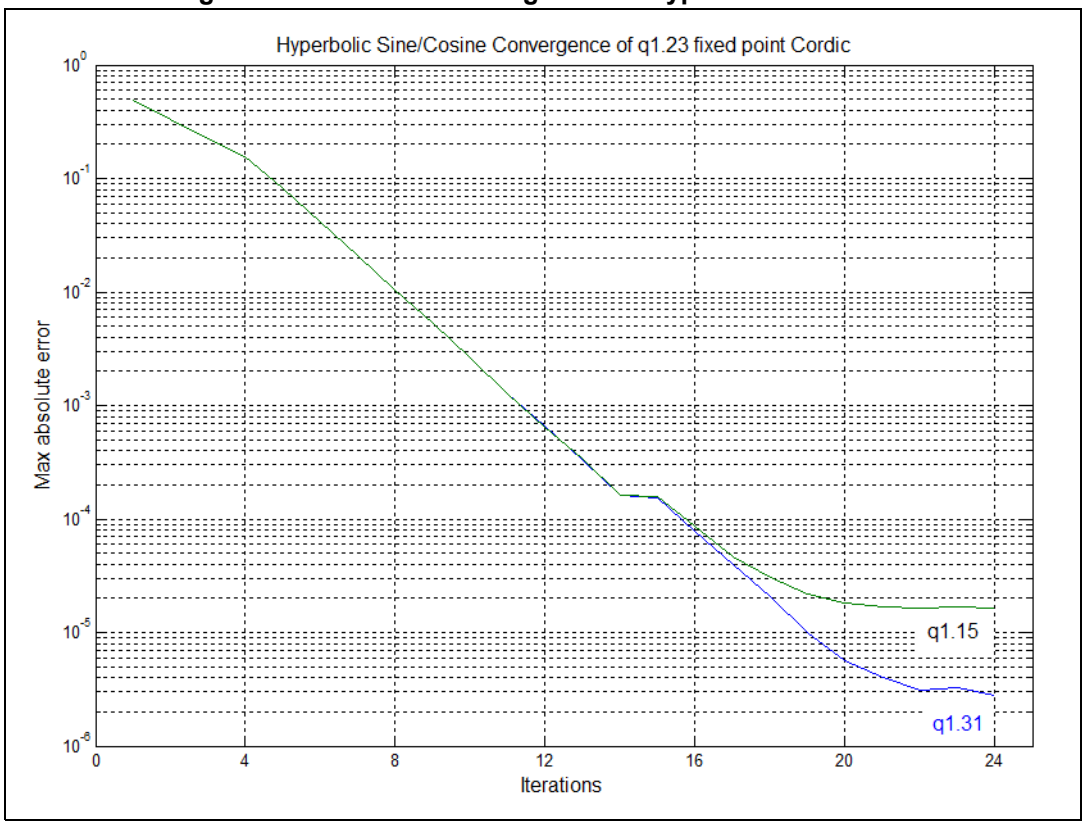
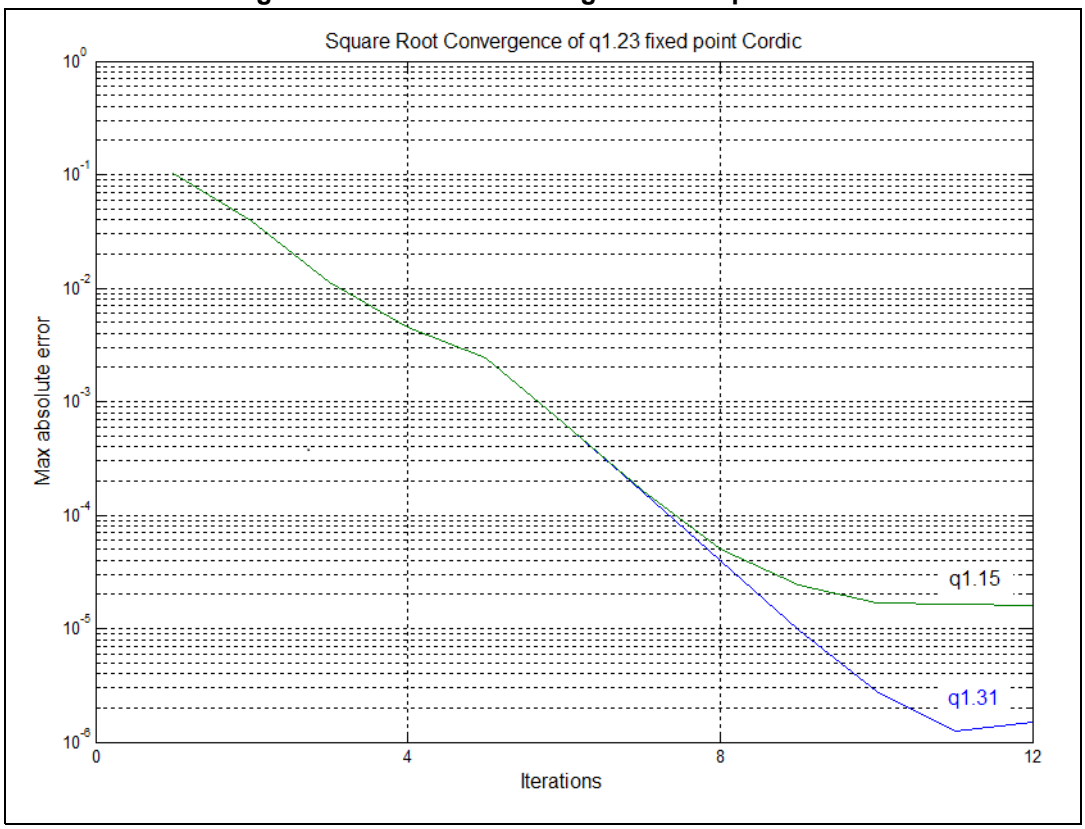


Figure 142. CORDIC convergence for square root



Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in *Table 196*, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The coprocessor stops as soon as the programmed number of iterations is completed, and the result can be read immediately.

Table 196. Precision vs. number of iterations

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sin, Cos, Phase(2), Mod, Atan(4)	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}

Table 196. Precision vs. number of iterations (continued)

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sinh, Cosh, Atanh, Ln(3)	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}
Sqrt(4)	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

1. Max residual error is the maximum error remaining after the given number of iterations, compared to the identical calculation performed in double precision floating point. An additional rounding error may be incurred, of up to 2^{-16} for q15 format or 2^{-20} for q31 format.
2. For modulus > 0.5. The achievable precision reduces proportionally to the magnitude of the modulus, as quantization error becomes significant.
3. SCALE = 1. If a higher scaling factor is used, the achievable precision is reduced proportionally.
4. SCALE = 0. If a higher scaling factor is used, the achievable precision is reduced proportionally.

21.3.6 Zero-overhead mode

The fastest way to use the coprocessor is to preprogram the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

The calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished), a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to reprogram the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument (ARG1) must be written first, followed by the secondary argument (ARG2). If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by reprogramming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions taking only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.

Once the calculation starts, any attempt to read the CORDIC_RDATA register inserts bus wait-states until the calculation is completed, before returning the result. It is then possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation, and prepare the next input data, if needed. The CORDIC_CSR register can be reprogrammed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous ones have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. IF NRES = 1, a second read of CORDIC_RDATA must be performed to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments has been written. This means that at any time, there can be a calculation in progress, or waiting for the results to be read, and an operation pending. Any further access to CORDIC_WDATA while an operation is pending cancels it and overwrites the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:

1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step *3*.

21.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice, depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. The processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (for example, if low latency interrupts must be serviced).

21.3.8 Interrupt mode

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However, it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

21.3.9 DMA mode

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention. This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. The sequence is started by the processor setting the DMAWEN flag, the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generates a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

Note: No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore useful only when repeatedly performing the same function with the same settings. The scale factor cannot be changed during a series of DMA transfers.

Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and may block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

21.4 CORDIC registers

The CORDIC registers can be accessed only in 32-bit word format

21.4.1 CORDIC control/status register (CORDIC_CSR)

Address offset: 0x00

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARG SIZE	RES SIZE	N ARGS	NRES	DMA WEN	DMA REN	IEN
r									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SCALE[2:0]			PRECISION[3:0]				FUNC[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RRDY**: Result ready flag

0: No new data in output register

1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.

When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **ARGSIZE**: Width of input data

0: 32-bit

1: 16-bit

ARGSIZE selects the number of bits used to represent input data.

If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.

If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format. The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 **RESSIZE**: Width of output data

0: 32-bit

1: 16-bit

RESSIZE selects the number of bits used to represent output data.

If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.

If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.

Bit 20 **NARGS**: Number of arguments expected by the CORDIC_WDATA register
0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.
1: Two 32-bit values must be written to the CORDIC_WDATA register to trigger the next calculation.
Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC_RDATA register
0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC_RDATA register on completion of the next calculation. One read from CORDIC_RDATA resets the RRDY flag.
1: Two 32-bit values are transferred to the CORDIC_RDATA register on completion of the next calculation. Two reads from CORDIC_RDATA are necessary to reset the RRDY flag.
Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel
0: Disabled. No DMA write requests are generated.
1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel
0: Disabled. No DMA read requests are generated.
1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt.
0: Disabled. No interrupt requests are generated.
1: Enabled. An interrupt request is generated whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SCALE[2:0]**: Scaling factor
The value of this field indicates the scaling factor applied to the arguments and/or results. A value n implies that the arguments have been multiplied by a factor 2^{-n}, and/or the results need to be multiplied by 2^{n}. Refer to *Section 21.3.2* for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)
0: reserved
1 to 15: (Number of iterations)/4
To determine the number of iterations needed for a given accuracy refer to *Table 196*.
Note that for most functions, the recommended range for this field is 3 to 6.

Bits 3:0 **FUNC[3:0]**: Function

0: Cosine
1: Sine
2: Phase
3: Modulus
4: Arctangent
5: Hyperbolic cosine
6: Hyperbolic sine
7: Arctanh
8: Natural logarithm
9: Square root
10 to 15: Reserved

21.4.2 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARG[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ARG[31:0]**: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to *Section 21.3.2* for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG 2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrite the argument data.

21.4.3 CORDIC result register (CORDIC_RDATA)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RES[31:0]**: Function result

If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.

If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.

If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed.

A read from this register resets the RRDY flag in the CORDIC_CSR register.

21.4.4 CORDIC register map

Table 197. CORDIC register map and reset value

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CORDIC_CSR	RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARGSIZE	RESSIZE	NARGS	NRES	DMAWEN	DMAREN	IEN	Res.	Res.	Res.	Res.	SCALE [2:0]				PRECISION [3:0]				FUNC [3:0]			
	Reset value	0									0	0	0	0	0	0	0					0	0	0	0	0	1	0	1	0	0	0	0
0x04	CORDIC_WDATA	ARG[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x08	CORDIC_RDATA	RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

22 Filter math accelerator (FMAC)

22.1 FMAC introduction

The filter math accelerator unit performs arithmetic operations on vectors. It comprises a multiplier/accumulator (MAC) unit, together with address generation logic which allows it to index vector elements held in local memory.

The unit includes support for circular buffers on input and output, which allows digital filters to be implemented. Both finite and infinite impulse response filters can be realized.

The unit allows frequent or lengthy filtering operations to be offloaded from the CPU, freeing up the processor for other tasks. In many cases it can accelerate such calculations compared to a software implementation, resulting in a speed-up of time critical tasks.

22.2 FMAC main features

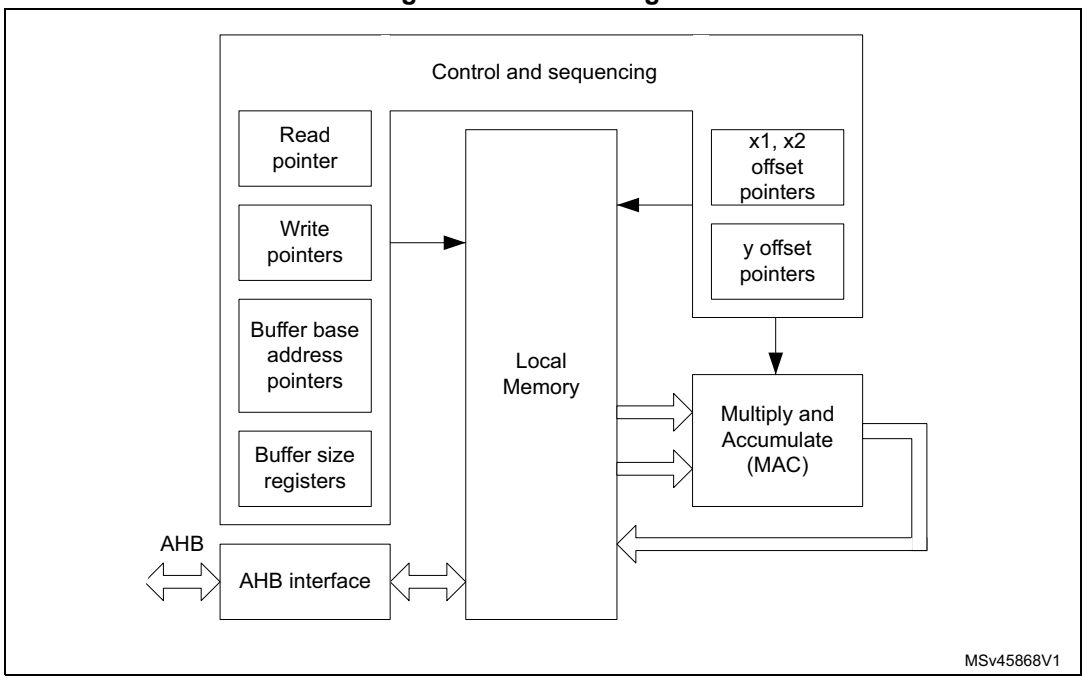
- 16 x 16-bit multiplier
- 24 + 2-bit accumulator with addition and subtraction
- 16-bit input and output data
- 256 x 16-bit local memory
- Up to three areas can be defined in memory for data buffers (two input, one output), defined by programmable base address pointers and associated size registers
- Input and output buffers can be circular
- Filter functions: FIR, IIR (direct form 1)
- Vector functions: Dot product, convolution, correlation
- AHB slave interface
- DMA read and write data channels

22.3 FMAC functional description

22.3.1 General description

The FMAC is shown in *Figure 143*.

Figure 143. Block diagram



The unit is built around a fixed point multiplier and accumulator (MAC). The MAC can take two 16-bit input signed values from memory, multiply them together and add them to the contents of the accumulator. The address of the input values in memory is determined using a set of pointers. These pointers can be loaded, incremented, decremented or reset by the internal hardware. The pointer and MAC operations are controlled by a built-in sequencer in order to execute the requested operation.

To calculate a dot product, the two input vectors are loaded into the local memory by the processor or DMA controller, and the requested operation is selected and started. Each pair of input vector elements is fetched from memory, multiplied together and accumulated. When all the vector elements have been processed, the contents of the accumulator are stored in the local memory, from where they can be read out by the processor or DMA.

The finite impulse response (FIR) filter operation (also known as convolution) consists in repeatedly calculating the dot product of the coefficient vector and a vector of input samples, the latter being shifted by one sample delay, with the least recent sample being discarded and a new sample added, at each repetition.

The infinite impulse response (IIR) filter operation is the convolution of the feedback coefficients with the previous output samples, added to the result of the FIR convolution.

A more detailed description of the filter operations is given in *Section 22.3.6: Filter functions*.

22.3.2 Local memory and buffers

The unit contains a 256 x 16-bit read/write memory which is used for local storage:

- Input values (the elements of the input vectors) are stored in two buffers, X1 and X2.
- Output values (the results of the operations) are stored in another buffer, Y.
- The locations and sizes of the buffers are designated as follows:
  - x1_base: the base address of the X1 buffer
  - x2_base: the base address of the X2 buffer
  - y_base: the base address of the Y buffer
  - x1_buf_size: the number of 16-bit addresses allocated to the X1 buffer
  - x2_buf_size: the number of 16-bit addresses allocated to the X2 buffer
  - y_buf_size: the number of 16-bit addresses allocated to the Y buffer.

These parameters are programmed in the corresponding registers when configuring the unit.

The CPU (or DMA controller) can initialize the contents of each buffer using the Initialization functions (*Section 22.3.5: Initialization functions*) and writing to the write data register. The data is transferred to the location within the target buffer indicated by a write pointer. After each new write, the write pointer is incremented. When the write pointer reaches the end of the allocated buffer space, it wraps back to the base address. This feature is used to load the elements of a vector prior to an operation, or to initialize a filter and load filter coefficients.

Buffer configuration

The buffer sizes and base address offsets must be configured in the X1, X2 and Y buffer configuration registers. For each function, the required buffer size is specified in the function description in *Section 22.3.6: Filter functions*. The base addresses can be chosen anywhere in internal memory, provided that all buffers fit within the internal memory address range (0x00 to 0xFF), that is, base address + buffer size must be less than 256.

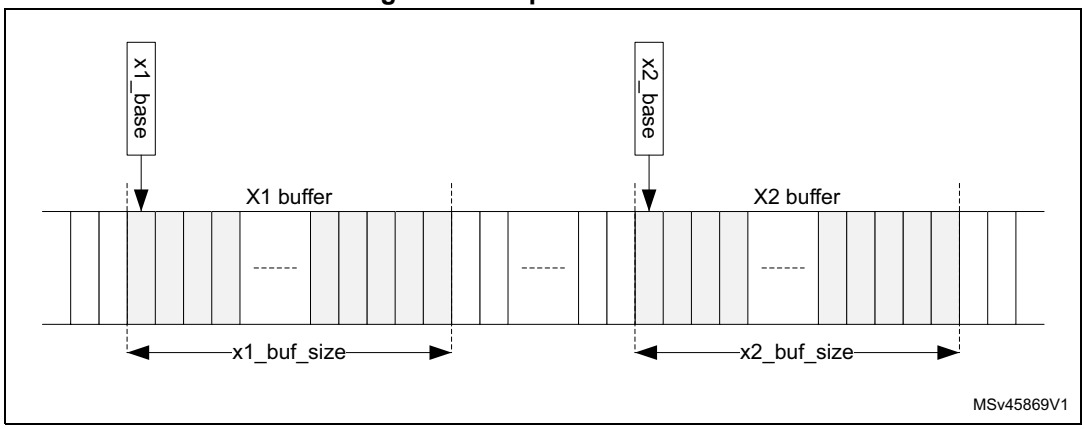
There is no constraint on the size and location of the buffers (they can overlap or even coincide exactly). For filter functions it is recommended not to overlap buffers as this can lead to erroneous behavior.

When circular buffer operation is required, an optional “headroom”, d, can be added to the buffer size. Furthermore, a watermark level can be set, to regulate the CPU or DMA activity. The value of d and the watermark level must be chosen according to the application performance requirements. For maximum throughput, the input buffer must never go empty, so d must be somewhat greater than the watermark level, allowing for any interrupt or DMA latency. On the other hand, if the input data can not be provided as fast as the unit can process them, the buffer can be allowed to empty waiting for the next data to be written, so d can be equal to the watermark level (to ensure that no overflow occurs on the input).

22.3.3 Input buffers

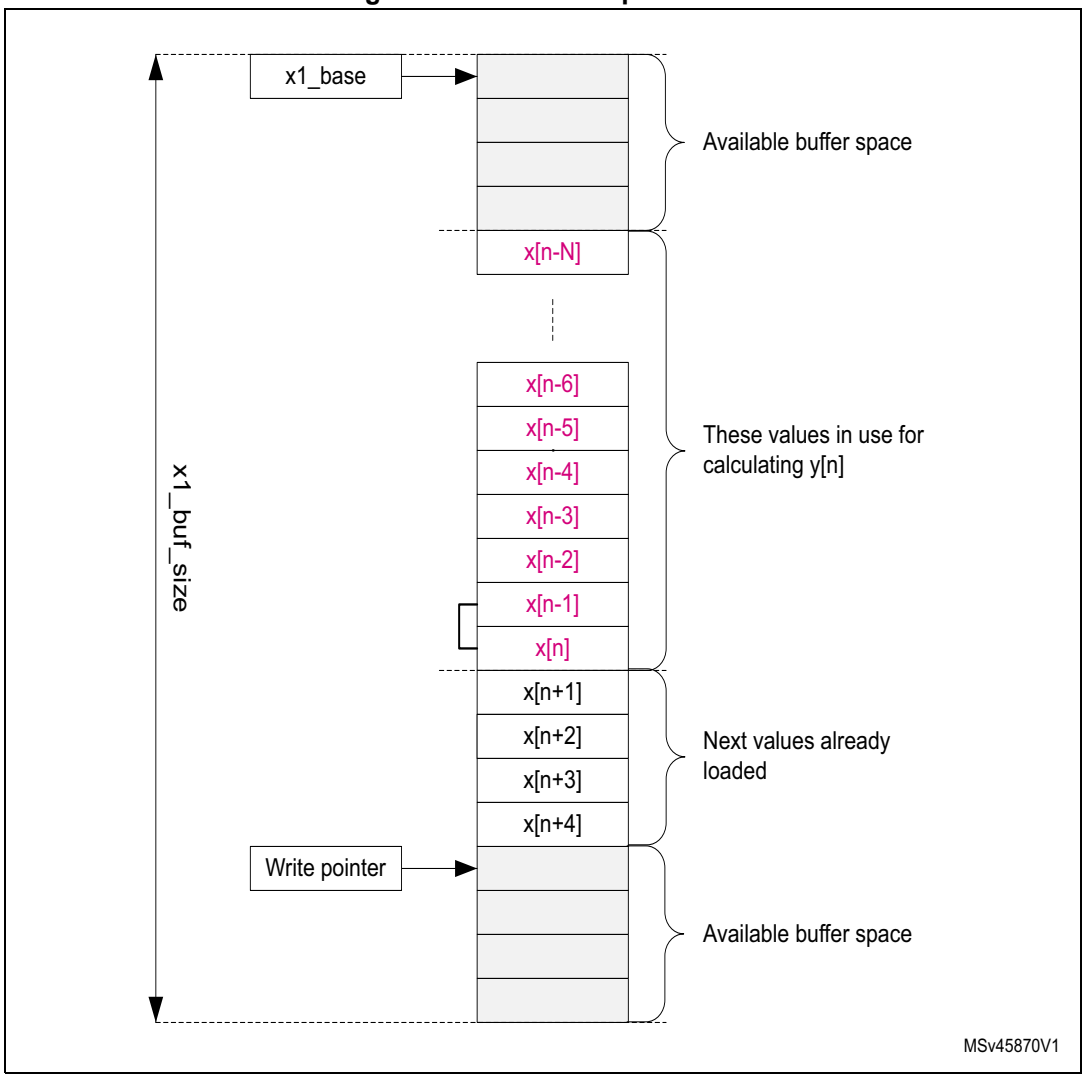
The X1 and X2 buffers are used to store data for input to the MAC. Each multiplication takes a value from the X1 buffer and a value from the X2 buffer and multiplies them together. A pointer in the control unit generates the read address offset (relative to the buffer base address) for each value. The pointers are managed by hardware according to the current function.

Figure 144. Input buffer areas



The X1 buffer can be used as a circular buffer, in which case new data are continually transferred into the input buffer whenever space is available. Pre-loading this buffer is optional for digital filters, since if no input samples have been written in the buffer when the operation is started, it is flagged as empty, which triggers the CPU or DMA to load new samples until there are enough to begin operation. Pre-loading is nevertheless useful in the case of a vector operation, that is, the input data is already available in system memory and circular operation is not required.

Figure 145. Circular input buffer



The X2 buffer can only be used in vector mode (that is not circular), and needs to be pre-loaded, except if the contents of the buffer do not change from one operation to the next. For filter functions, the X2 buffer is used to store the filter coefficients.

When operating as a circular buffer, the space allocated to the buffer (x1_buf_size) must generally be bigger than the number of elements in use for the current calculation, so that there are always new values available in the buffer. *Figure 145* illustrates the layout of the buffer for a filter operation. While calculating an output sample y[n], the unit uses a set of N+1 input samples, x[n-N] to x[n]. When this is finished, the unit starts the calculation of y[n+1], using the set of input samples x[n-N+1] to x[n+1]. The least-recent input sample, x[n-N], drops out of the input set, and a new sample, x[n+1], is added to it.

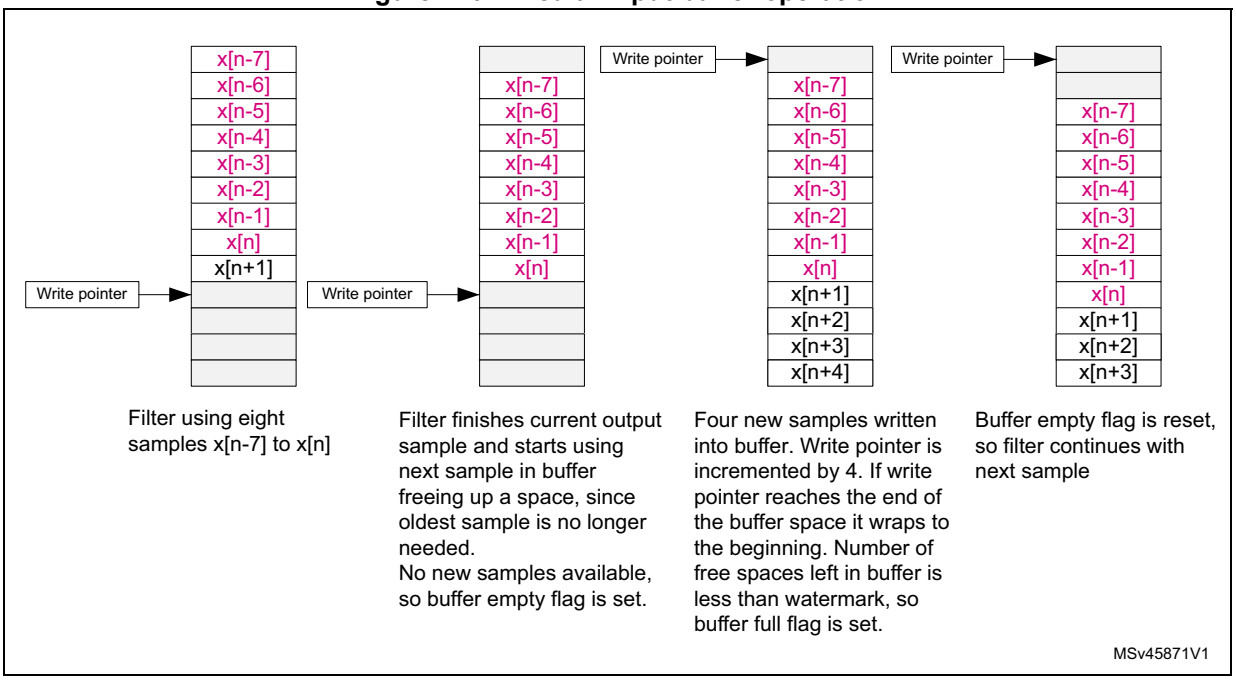
The processor, or DMA controller, must ensure that the new sample x[n+1] is available in the buffer space when required. If not, the buffer is flagged as empty, which stalls the execution of the unit until a new sample is added. No underflow condition is signaled on the X1 buffer.

Note: If the flow of samples is controlled by a timer or other peripheral such as an ADC, the buffer regularly goes empty, since the filter processes each new sample faster than the source can provide it. This is an essential feature of filter operation.

If the number of free spaces in the buffer is less than the watermark threshold programmed in the FULL_WM bitfield of the FMAC_X1BUFCFG register, the buffer is flagged as full. As long as the full flag is not set, interrupts are generated, if enabled, to request more data for the buffer. The watermark allows several data to be transferred under one interrupt, without danger of overflow. Nevertheless, if an overflow does occur, the OVFL error flag is set and the write data is ignored. The write pointer is not incremented in the event of an overflow.

The operation of the X1 buffer during a filtering operation is illustrated in *Figure 146*. This example shows an 8-tap FIR filter with a watermark set to four.

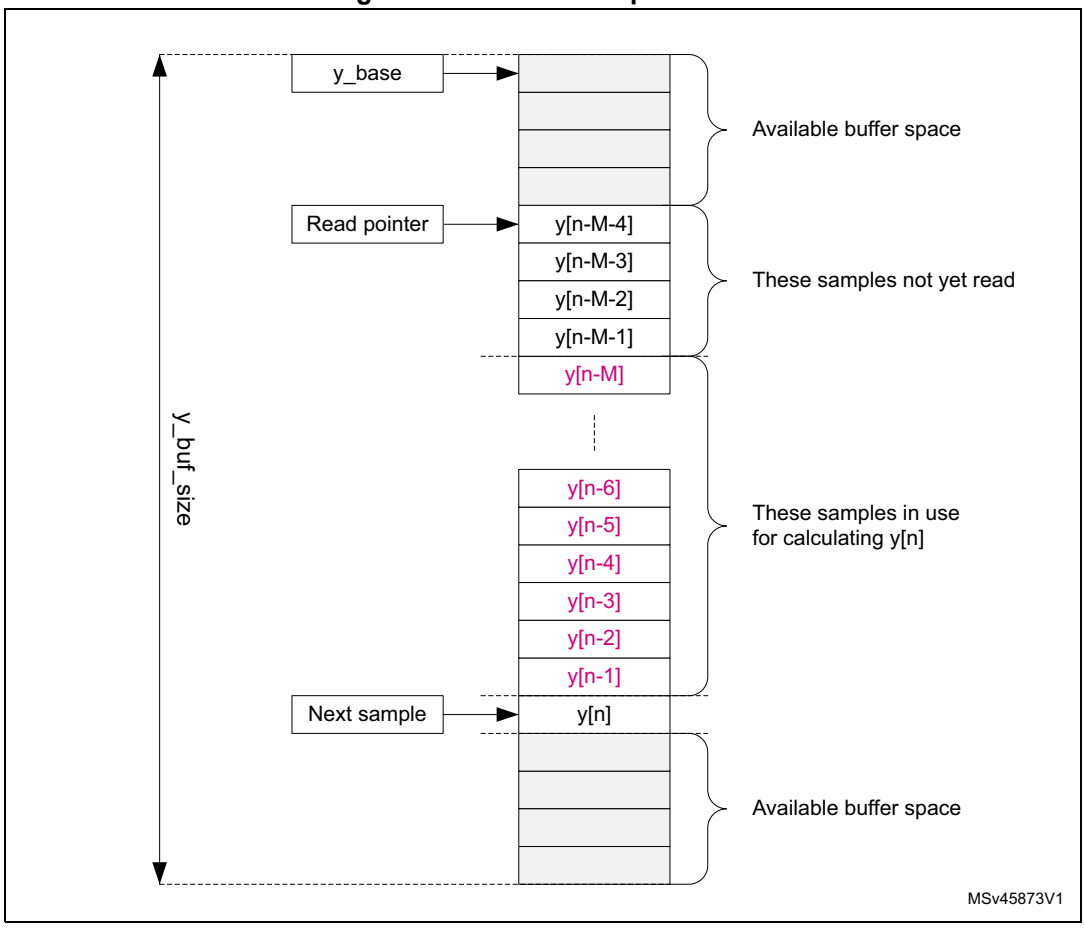
Figure 146. Circular input buffer operation



22.3.4 Output buffer

The Y (output) buffer is used to store the output of an accumulation. Each new output value is stored in the buffer until it is read by the processor or DMA controller. Each time a read access is made to the read data register, the read data is fetched from the address indicated by the read pointer. This pointer is incremented after each read, and wraps back to the base address when it reaches the end of the allocated Y buffer space.

Figure 147. Circular output buffer



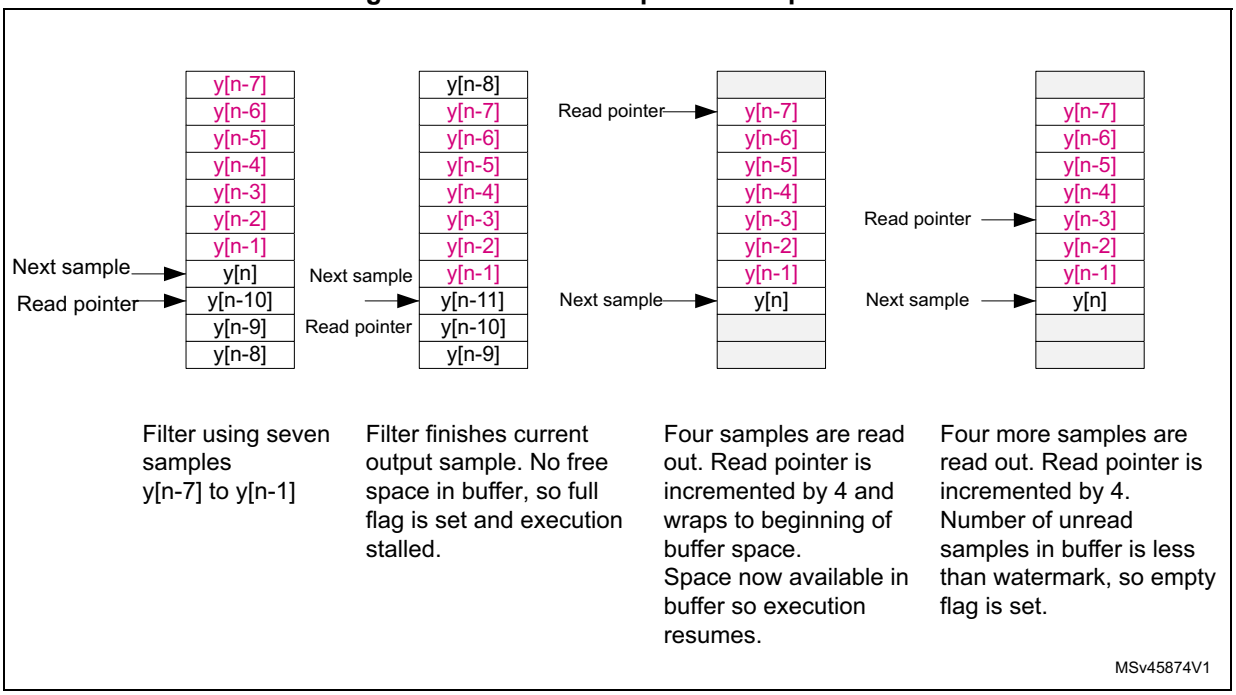
The Y buffer can also operate as a circular buffer. If the address for the next output value is the same as that indicated by the read pointer (an unread sample), then the buffer is flagged as full and execution stalled until the sample is read.

In the case of IIR filters, the Y buffer is used to store the set of M previous output samples, y[n-M] to y[n-1], used for calculating the next output sample y[n]. Each time a new sample is added to the set, the least recent sample y[n-M] drops out.

If the number of unread data in the buffer is less than the watermark threshold programmed in the EMPTY_WM bitfield of the FMAC_YBUFCFG register, the buffer is flagged as empty. As long as the empty flag is not set, interrupts or DMA requests are generated, if enabled, to request reads from the buffer. The watermark allows several data to be transferred under one interrupt, without danger of underflow. Nevertheless, if an underflow does occur, the UNFL error flag is set. In this case, the read pointer is not incremented and the read operation returns the content of the memory at the read pointer address.

The operation of the Y buffer in circular mode is illustrated in *Figure 148*. This example shows a 7-tap IIR filter with a watermark set to four.

Figure 148. Circular output buffer operation



22.3.5 Initialization functions

The following functions initialize the FMAC unit. They are triggered by writing the appropriate value in the FUNC bitfield of the FMAC_PARAM register, with the START bit set. The P and Q bitfields must also contain the appropriate parameter values for each function as detailed below. The R bitfield is not used. When the function completes, the START bit is automatically reset by hardware.

During initialization, it is recommended that the DMA requests and interrupts be disabled. The transfer of data into the FMAC memory can be done by software or by memory-to-memory DMA transfers, since no flow control is required.

Load X1 buffer

This function pre-loads the X1 buffer with N values, starting from the address in X1_BASE. Successive writes to the FMAC_WDATA register load the write data into the X1 buffer and increment the write address. The write pointer points to the address X1_BASE + N when the function completes.

The function can be used to pre-load the buffer with the elements of a vector, or to initialize the input storage elements of a filter.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X1 buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

Load X2 buffer

This function pre-loads the X2 buffer with N + M values, starting from the address in X2_BASE. Successive writes to the FMAC_WDATA register load the write data into the X2 buffer and increment the write address.

The function can be used to pre-load the buffer with the elements of a vector, or the coefficients of a filter. In the case of an IIR, the N feed-forward and M feed-back coefficients are concatenated and loaded together into the X2 buffer. The total number of coefficients is equal to N + M. For an FIR, there are no feedback coefficients, so M = 0.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X2 buffer starting from address X2_BASE.
- The parameter Q contains the number of values, M, to be loaded into the X2 buffer starting from address X2_BASE + N.
- The parameter R is not used.

The function completes when N + M writes have been performed to the FMAC_WDATA register.

Load Y buffer

This function pre-loads the Y buffer with N values, starting from the address in Y_BASE. Successive writes to the FMAC_WDATA register load the write data into the Y buffer and increment the write address. The read pointer points to the address Y_BASE + N when the function completes.

The function can be used to pre-load the feedback storage elements of an IIR filter.

Parameters

- The parameter P contains the number of values to be loaded into the Y buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

22.3.6 Filter functions

The following filter functions are supported by the FMAC unit. These functions are triggered by writing the corresponding value in the FUNC bitfield of the FMAC_PARAM register with the START bit set. The P, Q and R bitfields must also contain the appropriate parameter values for each function as detailed below. The filter functions continue to run until the START bit is reset by software.

Convolution (FIR filter)

Y = **B***__X__

$$y_n = 2^R \cdot \sum_{k=0}^{N} b_k x_{n-k}$$

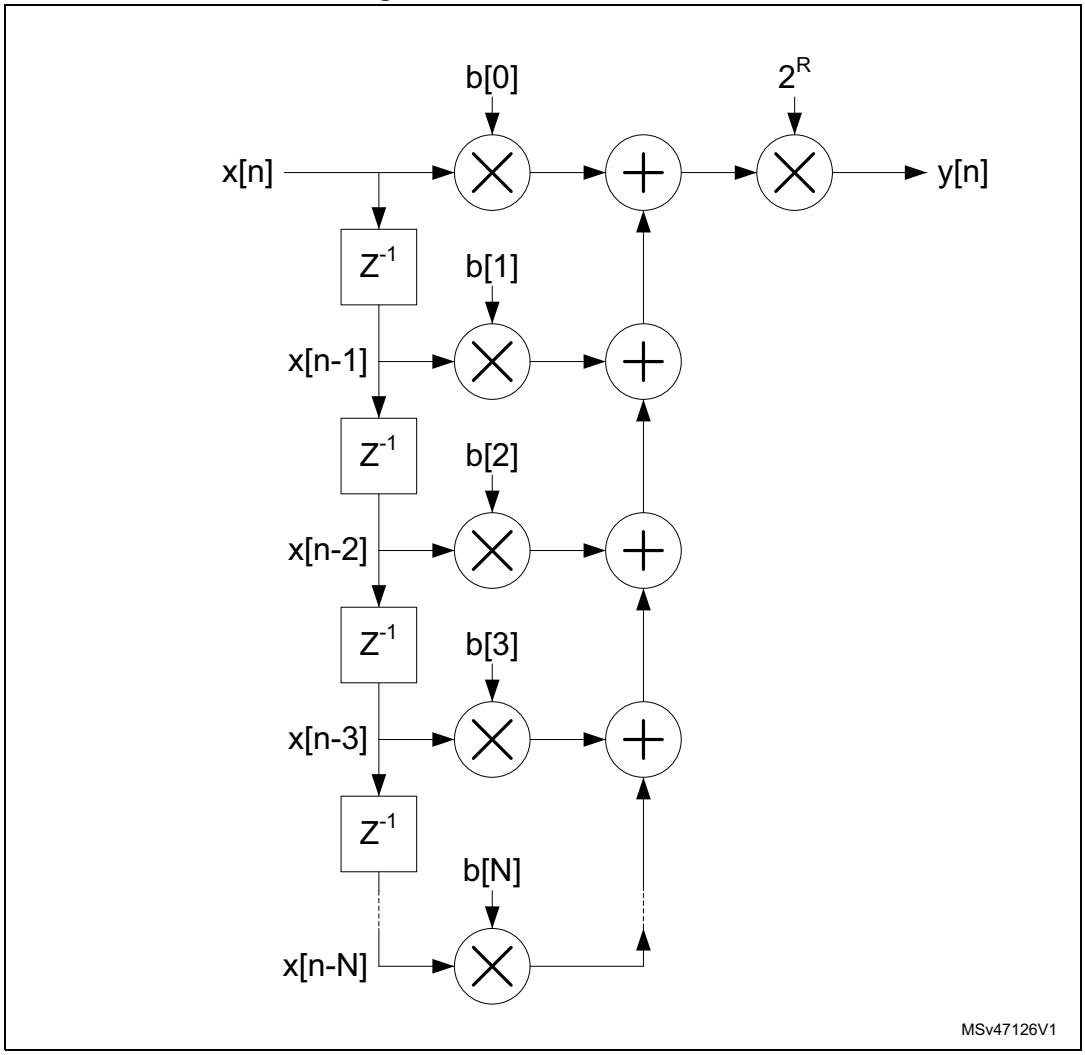
This function performs a convolution of a vector **B** of length N+1 and a vector **X** of indefinite length. The elements of **Y** for incrementing values of n are calculated as the dot product,

$y_n = \underline{\mathbf{B}}.\underline{\mathbf{X}}_{\mathbf{n}}$, where $\underline{\mathbf{X}}_{\mathbf{n}} = [x_{n-N},...,x_n]$ is composed of the N+1 elements of **X** at indexes n - N to n.

This function corresponds to a finite impulse response (FIR) filter, where vector **B** contains the filter coefficients and vector **X** the sampled data.

The structure of the filter (direct form) is shown in *Figure 149*.

Figure 149. FIR filter structure



Note that the cross correlation vector can be calculated by reversing the order of the coefficient vector **B**.

Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1 + d.
- X2 buffer contains the elements of vector **B**. It is a fixed buffer of length N + 1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length d.

Parameters:

- The parameter P contains the length, N+1, of the coefficient vector **B** in the range [2:127].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7]
- The parameter Q is not used.

The function completes when the START bit in the FMAC_PARAM register is reset by software.

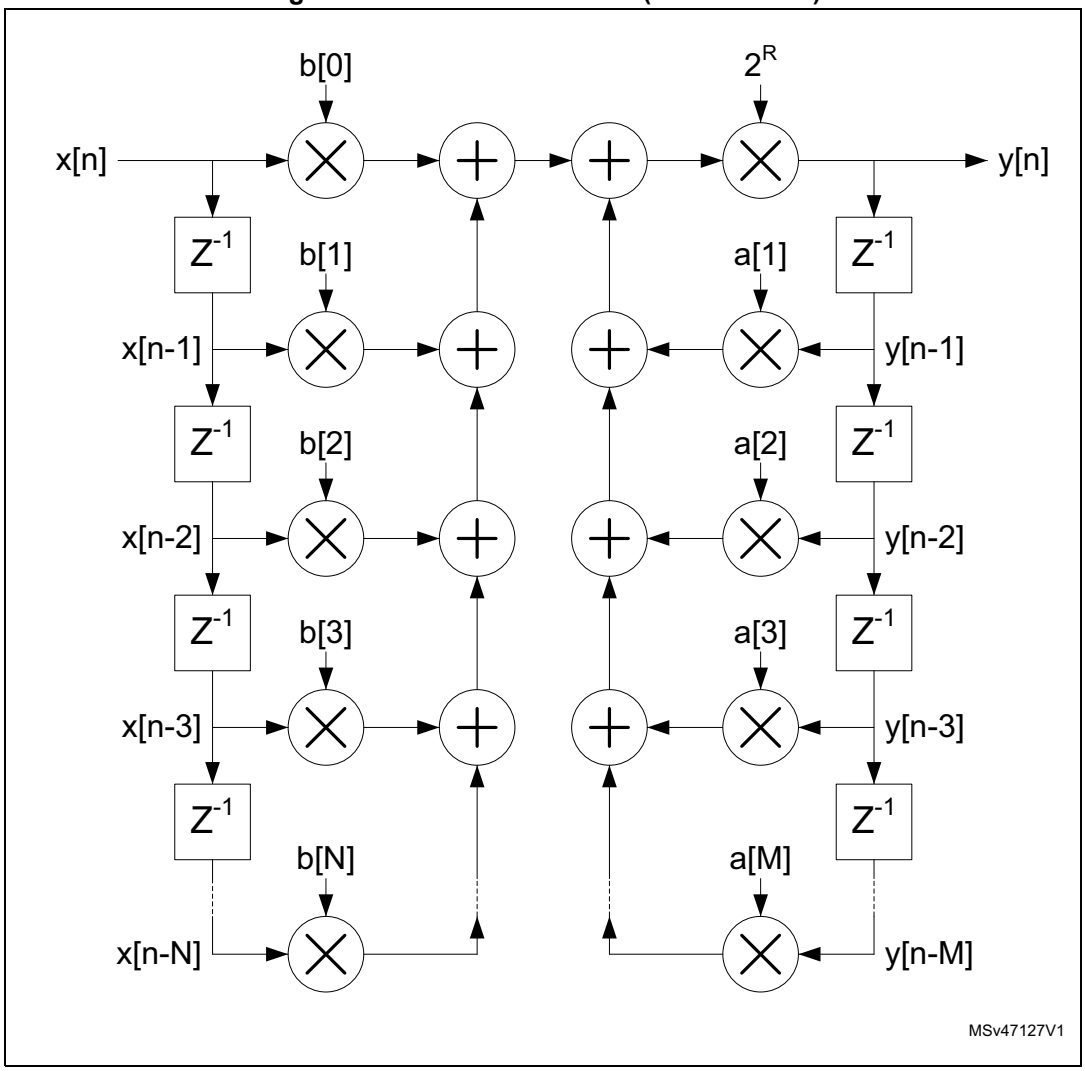
IIR filter

Y = B*X+A*Y'

$$y_n = 2^R \cdot \left(\sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right)$$

This function implements an infinite impulse response (IIR) filter. The filter output vector **Y** is the convolution of a coefficient vector **B** of length N+1 and a vector **X** of indefinite length, plus the convolution of the delayed output vector **Y'** with a second coefficient vector **A**, of length M. The elements of **Y** for incrementing values of n are calculated as y_n = **B**.$\mathbf{X_n}$ + **A**.$\mathbf{Y_{n-1}}$, where $\mathbf{X_n}$ = $[x_{n-N},...,x_n]$ comprises the N+1 elements of **X** at indexes n - N to n, while $\mathbf{Y_{n-1}}$ = $[y_{n-M},...,y_{n-1}]$ comprises the M elements of **Y** at indexes n - M to n - 1. The structure of the filter (direct form 1) is shown in *Figure 150*.

Figure 150. IIR filter structure (direct form 1)



Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1+ d.
- X2 buffer contains the elements of coefficient vectors **B** and **A** concatenated (b_0, b_1, b_2..., b_N, a_1, a_2, ..., a_M). It is a fixed buffer of length M+N+1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length M + d.

Parameters

- The parameter P contains the length, N + 1, of the coefficient vector **B** in the range [2:64].
- The parameter Q contains the length, M, of the coefficient vector **A** in the range [1:63].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7].

The function completes when the START bit in the FMAC_PARAM register is reset by software.

22.3.7 Fixed point representation

The FMAC operates in fixed point signed integer format. Input and output values are q1.15.

In q1.15 format, numbers are represented by one sign bit and 15 fractional bits (binary decimal places). The numeric range is therefore -1 (0x8000) to 1 - 2^{-15} (0x7FFF).

The accumulator has 26 bits, of which 22 are fractional and 4 are integer/sign (q4.22). This allows it to support partial accumulation sums in the range -8 (0x2000000) to +7.99999976 (0x1FFFFFF). A programmable gain from 0dB to 42dB in steps of 6dB can be applied at the output of the accumulator.

Note that the content of the accumulator is not saturated if the numeric range is exceeded. Partial sums whose value is greater than +7.99999976 or less than -8, wrap but this is harmless provided subsequent accumulations undo the wrapping. Nevertheless, the SAT flag in the FMAC_SR register is set if wrapping occurs, and generates an interrupt if the SATIEN bit is set in the FMAC_CR register. This helps in debugging the filter.

The data output by the accumulator can optionally be saturated, after application of the programmable gain, by setting the CLIPEN bit in the FMAC_CR register. If this bit is set, then any value which exceeds the numeric range of the q1.15 output, is set to 1 - 2^{-15} or -1, according to the sign. If clipping is not enabled, the unused accumulator bits after applying the gain is simply truncated.

22.3.8 Implementing FIR filters with the FMAC

The FMAC supports FIR filters of length N, where N is the number of taps or coefficients. The minimum local memory requirement for a FIR filter of length N is 2N + 1:

- N coefficients
- N input samples
- 1 output sample

Since the local memory size is 256, the maximum value for N is 127.

If maximum throughput is required, it may be necessary to allocate a small amount of extra space, d1 and d2, to the input and output sample buffers respectively, to ensure that the filter never stalls waiting for a new input sample, or waiting for the output sample to be read. In this case, the local memory requirement is 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N;
- Y_BUF_SIZE = d2 (or 1 if no extra space is required)

The buffer base addresses can be allocated anywhere, but the X2 buffer must not overlap with the others, or else the coefficients are overwritten. An example configuration is:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = 2N + d1

However, if the memory space is limited, the X1 and Y buffer areas can be overlapped, such that each output sample takes the place of the oldest input sample, which is no longer required:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = N

In this case, Y_BUF_SIZE = X1_BUF_SIZE = N + d1, so that the buffers remain in sync.

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $log_2(d2)$.

The filter coefficients **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. There is no point in pre-loading the Y buffer, since for the FIR filter there is no feedback path.

After configuring and initializing the buffers, the FMAC_CR register must be programmed according to the method used for writing and reading data to and from the FMAC memory.

Three methods are supported:

- Polling: No DMA request or Interrupt request is generated. Software must check that the X1_FULL flag is low before writing to WDATA, or that the Y_EMPTY flag is low before reading from RDATA.
- Interrupt: The interrupt request is asserted while the X1_FULL flag is low, for writes, or when the Y_EMPTY flag is low, for reads.
- DMA: DMA requests are asserted on the DMA write channel while the X1_FULL flag is low, and on the read channel while the Y_EMPTY flag is low.

Different methods can be used for read and for write. However it is not recommended to use both interrupts and DMA requests for the same operation[(a)]. The valid combinations are listed in *Table 198*.

Table 198. Valid combinations for read and write methods

WIEN	RIEN	DMAWEN	DMAREN	Write	Read
0	0	0	0	Polling	Polling
0	1	0	0	Polling	Interrupt
1	0	0	0	Interrupt	Polling
1	1	0	0	Interrupt	Interrupt
0	0	0	1	Polling	DMA
0	0	1	0	DMA	Polling
0	0	1	1	DMA	DMA
0	1	1	0	DMA	Interrupt
1	0	0	1	Interrupt	DMA

a. If both interrupts and DMA requests are enabled then only DMA must perform the transfer.

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 8 (FIR filter);
- P = N (number of coefficients);
- Q = “Don’t care”;
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine must check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples).

When 2^{EMPTY_WM} output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine must check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high.

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

22.3.9 Implementing IIR filters with the FMAC

The FMAC supports IIR filters of length N, where N is the number of feed-forward taps or coefficients. The number of feedback coefficients, M, can be any value from 1 to N-1. Only direct form 1 implementations can be realized, so filters designed for other forms need to be converted.

The minimum memory requirement for an IIR filter with N feed-forward coefficients and M feed-back coefficients is 2N + 2M:

- N + M coefficients
- N input samples
- M output samples

If M = N-1, then the maximum filter length that can be implemented is N = 64.

As for the FIR, for maximum throughput, a small amount of additional space, d1 and d2, is allowed in the input and output buffer size respectively, making the total memory requirement 2M + 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N + M;
- Y_BUF_SIZE = M + d2;

The buffer base addresses can be allocated anywhere, but must not overlap. An example configuration is given below:

- X2_BASE = 0;
- X1_BASE = N + M;
- Y_BASE = 2N + M + d1;

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $log_2(d2)$.

The filter coefficients (N feed-forward followed by M feedback) **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. The Y buffer can optionally be pre-loaded with any number of values up to a maximum of M. This has the effect of initializing the feedback delay line.

After configuring the buffers, the FMAC_CR register must be programmed in the same way as for the FIR filter (see *Section 22.3.8: Implementing FIR filters with the FMAC*).

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 9 (IIR filter);
- P = N (number of feed-forward coefficients);
- Q = M (number of feed-back coefficients);
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine must check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

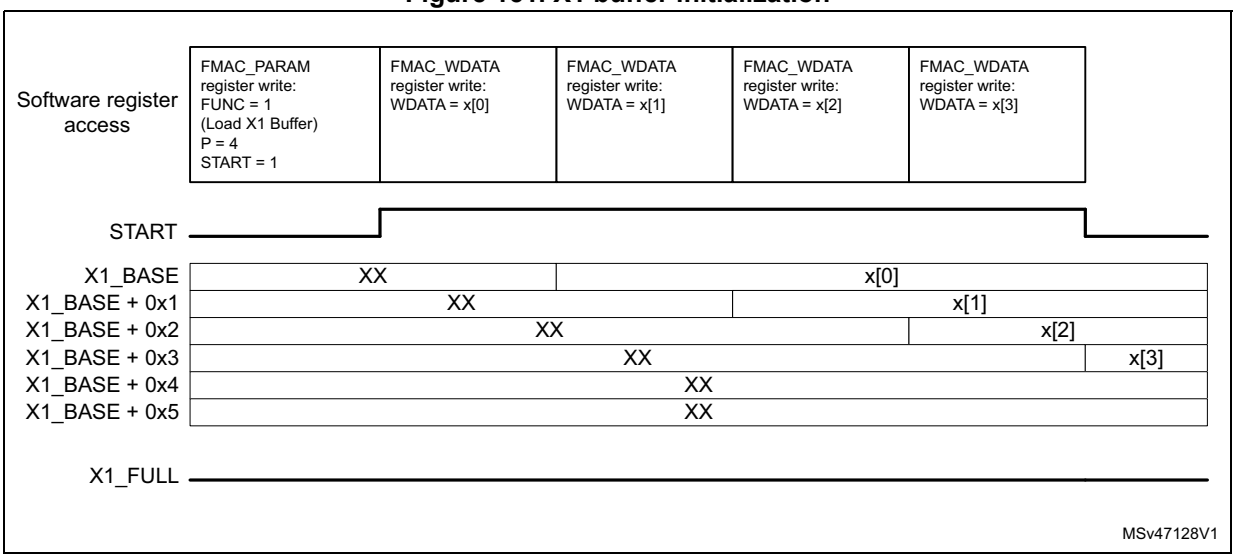
The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples). The first sample is calculated using the first N samples in the X1 buffer, and the first M samples in the Y buffer (whether or not they are preloaded. The first output sample is written into the Y buffer at Y_BASE + M.

When 2^{EMPTY_WM} new output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine must check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

22.3.10 Examples of filter initialization

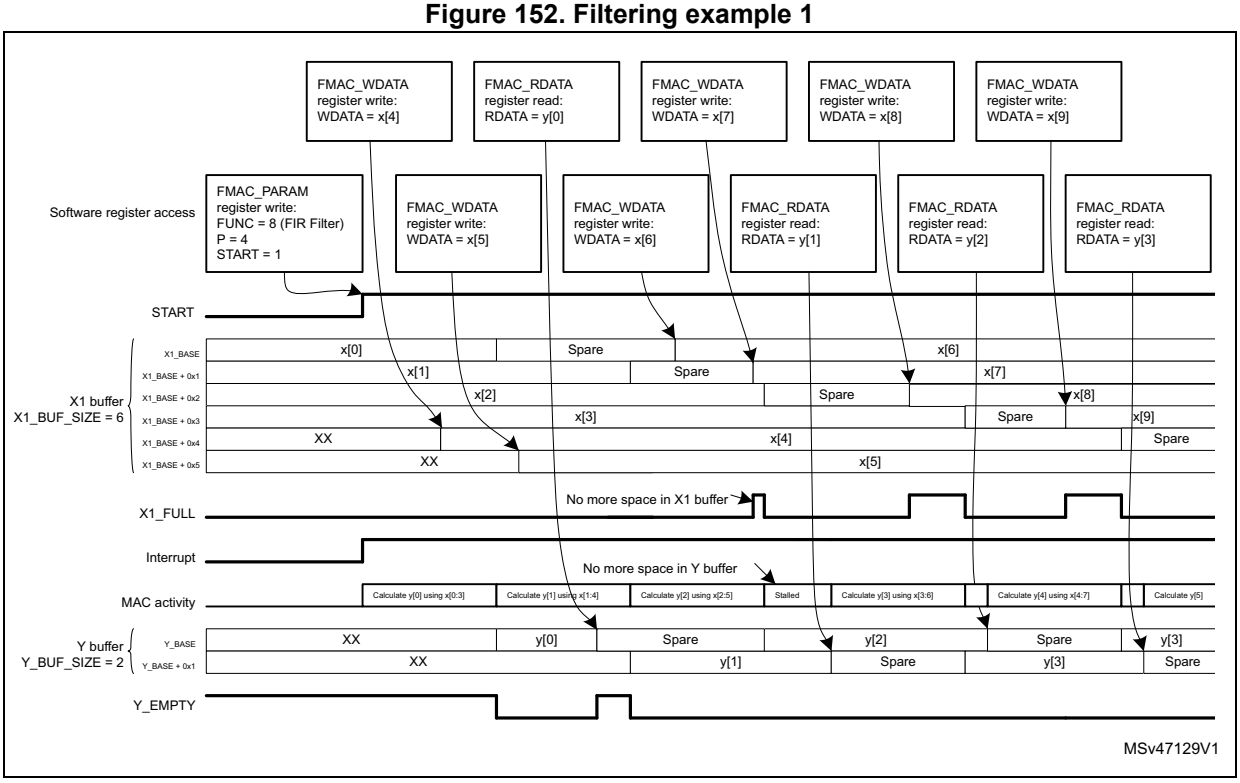
Figure 151. X1 buffer initialization



The example in *Figure 151* illustrates an X1 buffer pre-load with four samples (P = 4). The buffer size is six (X1_BUF_SIZE = 6). The initialization is launched by programming the FMAC_PARAM register with the START bit set. The four samples are then written to FMAC_WDATA, and transferred into local memory from X1_BASE onwards. The START bit resets after the fourth sample has been written. At this point, the X1 buffer contains the four samples, in order of writing, and the write pointer (next empty space) is at X1_BASE + 0x4.

22.3.11 Examples of filter operation

Figure 152. Filtering example 1



The example in *Figure 152* illustrates the beginning of a filter operation. The filter has four taps (P=4). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 151*. So the filter starts calculating the first output sample, y[0], immediately after the START bit is set. Since the X1FULL flag is not set (due to two uninitialized spaces in the X1 buffer), the interrupt is asserted straight away, to request new data. The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer.

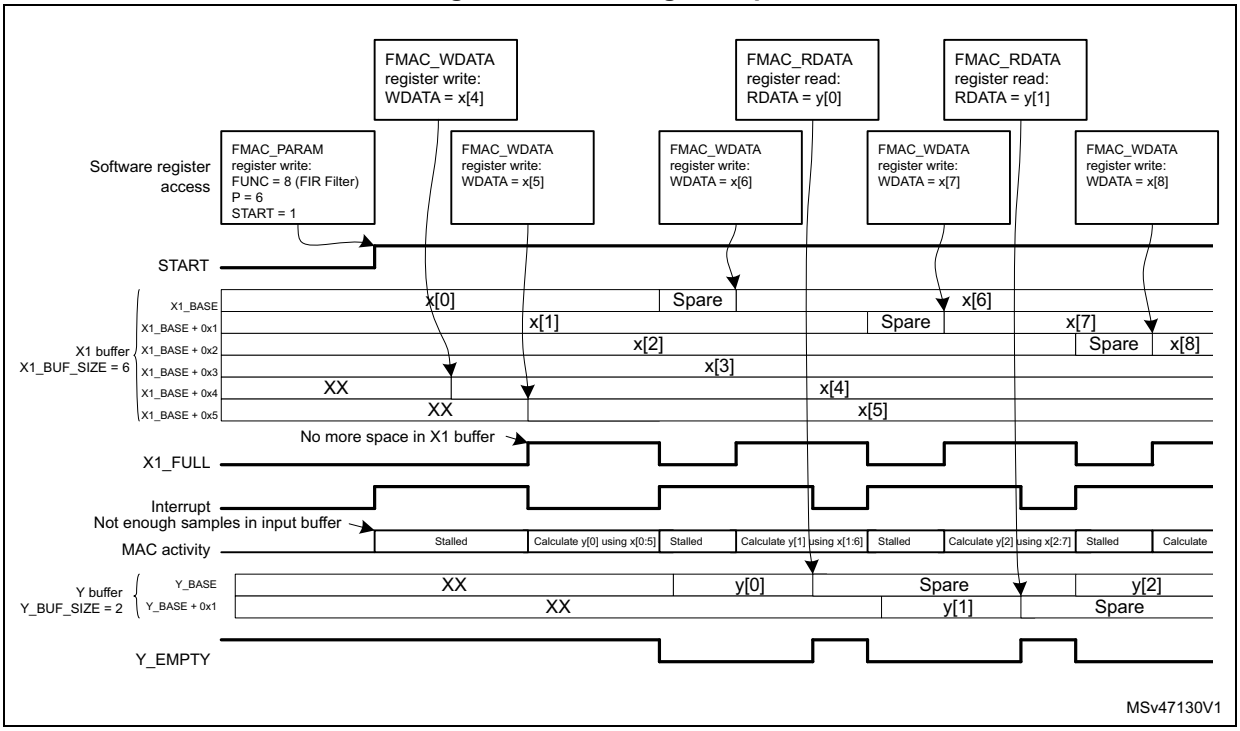
In the mean time, the FMAC finishes calculating the first output sample, y[0], and writes it into the Y buffer, causing the Y_EMPTY flag to go low. At the same time, the x[0] sample is discarded, as it is no longer required, freeing up its location in memory (at X1_BASE). The FMAC can immediately start work on the second output sample, y[1], since all the required input samples x[1:5] are present in the X1 buffer.

Since the Y_EMPTY flag is low, the interrupt remains active after the processor finishes writing x[5]. The processor reads y[0] from the FMAC_RDATA register, freeing up its location in the Y buffer. There are now no samples in the output buffer since y[1] is still being calculated, so the Y_EMPTY flag goes high. Nevertheless, the interrupt remains active, because there is still free space in the X1 buffer, which the processor next fills with x[6], and so on.

Note: In this example, the processor can fill the input buffer more quickly than the FMAC can process them, so the X1_full flag regularly goes active. However, it struggles to read the Y buffer fast enough, so the FMAC stalls regularly waiting for space to be freed up in the Y buffer. This means the filter is not executing at maximum throughput. The reason is that the

filter length is small and the processor relatively slow, in this example. So increasing the Y buffer size does not help.

Figure 153. Filtering example 2



The example in *Figure 153* illustrates the beginning of the same filter operation, but this time the filter has six taps (P=6). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 151*. Because there are not enough samples in the input buffer, the X1FULL flag is not set, so the interrupt is asserted straight away, to request new data. The FMAC is stalled.

The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer. As soon as there are six unused samples in the X1 buffer, the X1_FULL flag goes active (since the buffer size is six), causing the interrupt to go inactive. The FMAC starts calculating the first output sample, y[0]. Since this requires all six input samples, there are no free spaces in the X1 buffer and so the X1_FULL flag remains active. Only when the FMAC finishes calculating y[0] and writes it into the Y buffer, can x[0] be discarded, freeing up a space in the X1 buffer, and deasserting X1_FULL. At the same time, the Y_EMPTY flag goes inactive. Both these flag states cause the interrupt to be asserted, requesting the processor to write a new input sample, first of all, and then read the output sample just calculated. The FMAC remains stalled until a new input sample is written.

In this example, the processor has to wait for the FMAC to finish calculating the current output sample, before it can write a new input sample, and therefore the X1 buffer regularly goes empty, stalling the FMAC. This can be avoided by allowing some extra space in the input buffer.

22.3.12 Filter design tips

The FMAC architecture imposes some constraints detailed below, on the design of digital filters.

1. Implementation of direct form 2, or transposed forms, is not efficient. Filters which have been designed for such forms must be converted to direct form 1.
2. Cascaded filters must either be combined into a single stage, or implemented as separate filters. In the latter case, multiple sets of filter coefficients can be pre-loaded into the memory, one set per stage, and only the X2_BASE address changed to select which set is used. The most efficient method of implementing a multi-stage filter is to pre-load a large X1 buffer with input samples, run the IIR filter function on it using the first stage coefficients, and store the output samples back in memory. Then change the X2_BASE pointer to point to the 2nd stage coefficients, and reload the input buffer with the output of the first stage (with a gain if required), before running the IIR function again. The procedure is repeated for all stages. Once the final stage samples have been transferred back into system memory, the input buffer can be loaded with the next set of input samples, and a new round of calculations started. Note that the N sample input buffer of each stage must be pre-loaded first of all with the N-1 last inputs from the previous round, plus one new sample, in order to keep continuity between each round. Similarly, the output buffer of each stage must be loaded with the last M samples from the previous round, for the same reason.
3. The use of direct form 1 for IIR designs can lead to large positive or negative partial sums in the accumulator, if for example a large step occurs on the input, or some of the filter coefficients' absolute values are >1. Since the accumulator is limited to 26 bits, the biggest value that it can handle without wrapping (changing sign) is 0x1FFFFFF positive or 0x2000000 negative. This corresponds to 3.99999988 and -4 respectively in q3.23 fixed point format. Wrapping does not represent a problem provided the wrapping is "undone" before the end of the accumulation. However this is not always the case when a filter is starting up and can lead to unexpected results. Consider pre-loading the output buffer with suitable values to avoid this.
4. The IIR filter has feed-forward (numerator) coefficients $[b_0, b_1, ..., b_{N-1}]$, and feed-back (denominator) coefficients $[1, a_1, ..., a_M]$. Many IIR filters require some of the denominator coefficients to have an absolute value greater than 1 to achieve a steep roll-off in the frequency response. Given that the coefficients are coded in fixed point q1.15 format, this is not possible. Nevertheless, by scaling the denominator coefficients by a factor 2^{-R}, such that $2^{-R}.[1, a_1, ..., a_M]$ are all less than 1, such filters can be implemented. However an inverse gain of 2^R must be applied at the output of the accumulator to compensate the scaling. This has an adverse effect on the signal-to-noise ratio.

22.4 FMAC registers

22.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG)

Address offset: 0x00

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FULL_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **FULL_WM[1:0]**: Watermark for buffer full flag

Defines the threshold for setting the X1 buffer full flag when operating in circular mode. The flag is set if the number of free spaces in the buffer is less than 2^{FULL_WM}.

0: Threshold = 1

1: Threshold = 2

2: Threshold = 4

3: Threshold = 8

Setting a threshold greater than 1 allows several data to be transferred into the buffer under one interrupt.

Threshold must be set to 1 if DMA write requests are enabled (DMAWEN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **X1_BUF_SIZE[7:0]**: Allocated size of X1 buffer in 16-bit words

The minimum buffer size is the number of feed-forward taps in the filter (+ the watermark threshold - 1).

Bits 7:0 **X1_BASE[7:0]**: Base address of X1 buffer

22.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG)

Address offset: 0x04

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **X2_BUF_SIZE[7:0]**: Size of X2 buffer in 16-bit words
This bitfield can not be modified when a function is ongoing (START = 1).

Bits 7:0 **X2_BASE[7:0]**: Base address of X2 buffer
The X2 buffer base address can be modified while START=1, for example to change coefficient values. The filter must be stalled when doing this, since changing the coefficients while a calculation is ongoing affects the result.

22.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG)

Address offset: 0x08

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	EMPTY_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Y_BUF_SIZE[7:0]								Y_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **EMPTY_WM[1:0]**: Watermark for buffer empty flag
Defines the threshold for setting the Y buffer empty flag when operating in circular mode. The flag is set if the number of unread values in the buffer is less than 2^{EMPTY_WM}.
0: Threshold = 1
1: Threshold = 2
2: Threshold = 4
3: Threshold = 8
Setting a threshold greater than 1 allows several data to be transferred from the buffer under one interrupt.
Threshold must be set to 1 if DMA read requests are enabled (DMAREN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **Y_BUF_SIZE[7:0]**: Size of Y buffer in 16-bit words
For FIR filters, the minimum buffer size is 1 (+ the watermark threshold). For IIR filters the minimum buffer size is the number of feedback taps (+ the watermark threshold).

Bits 7:0 **Y_BASE[7:0]**: Base address of Y buffer

22.4.4 FMAC parameter register (FMAC_PARAM)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
START	FUNC[6:0]							R[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Q[7:0]								P[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **START**: Enable execution
0: Stop execution
1: Start execution
Setting this bit triggers the execution of the function selected in the FUNC bitfield. Resetting it by software stops any ongoing function. For initialization functions, this bit is reset by hardware.

Bits 30:24 **FUNC[6:0]**: Function
0: Reserved
1: Load X1 buffer
2: Load X2 buffer
3: Load Y buffer
4 to 7: Reserved
8: Convolution (FIR filter)
9: IIR filter (direct form 1)
10 to 127: Reserved
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 23:16 **R[7:0]**: Input parameter R.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 15:8 **Q[7:0]**: Input parameter Q.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 7:0 **P[7:0]**: Input parameter P.
The value of this parameter is dependent on the function
This bitfield can not be modified when a function is ongoing (START = 1)

22.4.5 FMAC control register (FMAC_CR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLIP EN	Res.	Res.	Res.	Res.	Res.	DMA WEN	DMA REN	Res.	Res.	Res.	SAT IEN	UNFL IEN	OVFL IEN	WIEN	RIEN
rw						rw	rw				rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RESET**: Reset FMAC unit

This resets the write and read pointers, the internal control logic, the FMAC_SR register and the FMAC_PARAM register, including the START bit if active. Other register settings are not affected. This bit is reset by hardware.

0: Reset inactive

1: Reset active

Bit 15 **CLIPEN**: Enable clipping

0: Clipping disabled. Values at the output of the accumulator which exceed the q1.15 range, wrap.

1: Clipping enabled. Values at the output of the accumulator which exceed the q1.15 range are saturated to the maximum positive or negative value (+1 or -1) according to the sign.

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **DMAWEN**: Enable DMA write channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the X1 buffer is not full.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bit 8 **DMAREN**: Enable DMA read channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the Y buffer is not empty.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SATIEN**: Enable saturation error interrupts

0: Disabled. No interrupts are generated upon saturation detection.

1: Enabled. An interrupt request is generated if the SAT flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 3 **UNFLIEN**: Enable underflow error interrupts

0: Disabled. No interrupts are generated upon underflow detection.

1: Enabled. An interrupt request is generated if the UNFL flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 2 **OVFLIEN**: Enable overflow error interrupts
0: Disabled. No interrupts are generated upon overflow detection.
1: Enabled. An interrupt request is generated if the OVFL flag is set
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 1 **WIEN**: Enable write interrupt
0: Disabled. No write interrupt requests are generated.
1: Enabled. An interrupt request is generated while the X1 buffer FULL flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 0 **RIEN**: Enable read interrupt
0: Disabled. No read interrupt requests are generated.
1: Enabled. An interrupt request is generated while the Y buffer EMPTY flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

22.4.6 FMAC status register (FMAC_SR)

Address offset: 0x14

Reset value: 0x0000 0001

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1 FULL	Y EMPTY
					r	r	r							r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SAT**: Saturation error flag
Saturation occurs when the result of an accumulation exceeds the numeric range of the accumulator.
0: No saturation detected
1: Saturation detected. If the SATIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 9 **UNFL**: Underflow error flag
An underflow occurs when a read is made from FMAC_RDATA when no valid data is available in the Y buffer.
0: No underflow detected
1: Underflow detected. If the UNFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 8 **OVFL**: Overflow error flag
An overflow occurs when a write is made to FMAC_WDATA when no free space is available in the X1 buffer.
0: No overflow detected
1: Overflow detected. If the OVFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **X1FULL**: X1 buffer full flag

The buffer is flagged as full if the number of available spaces is less than the FULL_WM threshold. The number of available spaces is the difference between the write pointer and the least recent sample currently in use.

0: X1 buffer not full. If the WIEN bit is set, the interrupt request is asserted until the flag is set. If DMAWEN is set, DMA write channel requests are generated until the flag is set.

1: X1 buffer full.

This flag is set and cleared by hardware, or by a reset.

Note: after the last available space in the X1 buffer is filled there is a delay of 3 clock cycles before the X1FULL flag goes high. To avoid any risk of overflow it is recommended to insert a software delay after writing to the X1 buffer before reading the FMAC_SR. Alternatively, a FULL_WM threshold of 2 can be used.

Bit 0 **YEMPTY**: Y buffer empty flag

The buffer is flagged as empty if the number of unread data is less than the EMPTY_WM threshold. The number of unread data is the difference between the read pointer and the current output destination address.

0: Y buffer not empty. If the RIEN bit is set, the interrupt request is asserted until the flag is set. If DMAREN is set, DMA read channel requests are generated until the flag is set.

1: Y buffer empty.

This flag is set and cleared by hardware, or by a reset.

Note: after the last sample is read from the Y buffer there is a delay of 3 clock cycles before the YEMPTY flag goes high. To avoid any risk of underflow it is recommended to insert a software delay after reading from the Y buffer before reading the FMAC_SR. Alternatively, an EMPTY_WM threshold of 2 can be used.

22.4.7 FMAC write data register (FMAC_WDATA)

Address offset: 0x18

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDATA[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WDATA[15:0]**: Write data

When a write access to this register occurs, the write data are transferred to the address offset indicated by the write pointer. The pointer address is automatically incremented after each write access.

22.4.8 FMAC read data register (FMAC_RDATA)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RDATA[15:0]**: Read data

When a read access to this register occurs, the read data are the contents of the Y output buffer at the address offset indicated by the READ pointer. The pointer address is automatically incremented after each read access.

22.4.9 FMAC register map

Table 199.FMAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMAC_X1BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	FULL_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	FMAC_X2BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	FMAC_YBUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Y_BUF_SIZE[7:0]								Y_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FMAC_PARAM	START	FUNC[6:0]							R[7:0]								Q[7:0]								P[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	FMAC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET	CLIPEN	Res.	Res.	Res.	Res.	Res.	DMAWEN	DMAREN	Res.	Res.	Res.	SATIEN	UNFLIEN	OVFLIEN	WIEN	RIEN
	Reset value																0	0						0	0				0	0	0	0	0
0x14	FMAC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1FULL	YEMPTY
	Reset value																						0	0	0							0	1
0x18	FMAC_WDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	FMAC_RDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

23 Programmable logic array (PLAY)

This section is applied only to STM32G55x/56x devices.

23.1 PLAY introduction

The programmable logic array allows the user to create custom logic and state machines without the need for external programmable logic devices, such as FPGAs. The PLAY provides a number of programmable logic elements, in the form of look-up tables (LUTs), each of which have four inputs and one output. The output can be programmed to be any logical function of the four inputs. More complex functions can be created by cascading logic elements.

Inputs to and outputs from the logic array can be selected from a choice of external I/Os, internal signals or software programmable register bits. External inputs can be synchronized to the internal clock and filtered to remove short pulses, or connected directly to the logic array if purely combinatorial or asynchronous functions are required.

Each logic element output can be registered using the internal clock, or another signal. Registered outputs can be fed back to the logic array inputs to create finite state machines. Transitions on the outputs can set flags, which can generate interrupts to the processor. The state of the flags can be read and reset by software. The raw states of the logic element outputs can also be read by software or by an external debugger. This provides visibility of the logic array state to simplify debugging of the programmed logic functions.

The array is configured via memory-mapped registers. Configuration is simple and intuitive. Configuration registers are protected against accidental programming by a lock bit. The configuration is maintained in case of a warm system reset.

23.2 PLAY main features

- Glitch-free programmable logic elements each comprising a 4-input look-up table and a register
- Configurable input and output interconnects
- Optional synchronization of inputs, with programmable glitch filters and edge detection/pulse extension feature.
- Software programmable inputs
- Flags with interrupt capability for communication with software
- Simple register-based configuration interface, protected by software lock

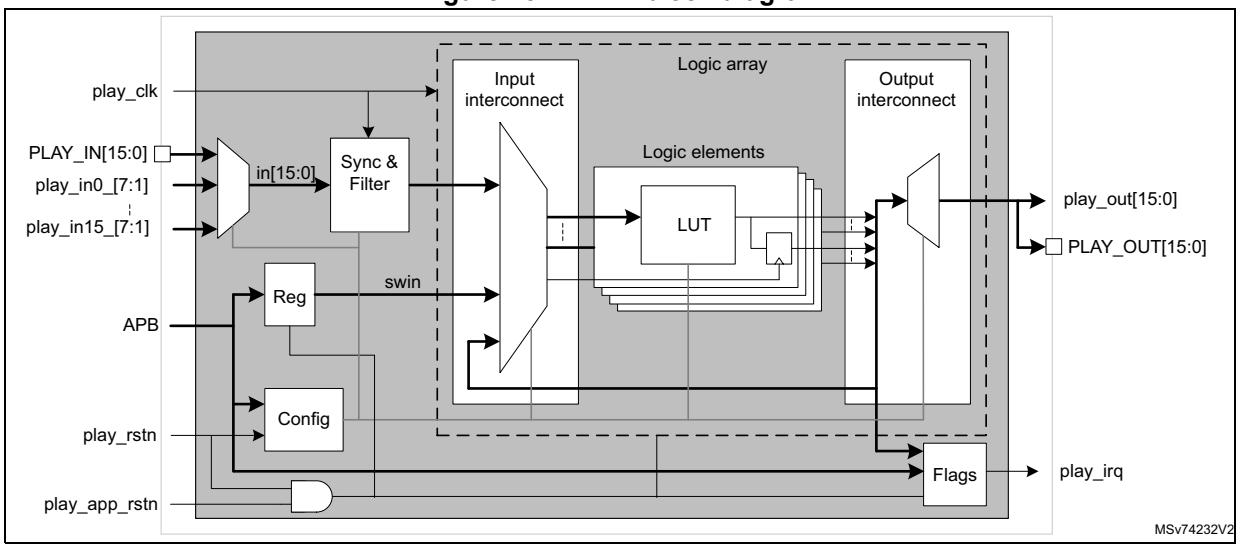
23.3 PLAY functional description

- a logic array, including
  - 16 logic elements
  - an input interconnect
  - an output interconnect
- a synchronization and filter unit
- a configuration unit
- a flag and interrupt unit
- an APB bus slave interface

These are shown in *Figure 154*.

23.3.1 PLAY block diagram

Figure 154. PLAY block diagram



23.3.2 PLAY pins and internal signals

Table 200. PLAY pins

Pin name	Pin type	Description
PLAY_IN[15:0]	Digital inputs	External inputs to the PLAY
PLAY_OUT[15:0]	Digital outputs	External outputs from the PLAY

Table 201. PLAY internal signals

Signal name	Signal type	Description
play_in[15:0]_[7:1]	Digital inputs	PLAY internal inputs
play_out[15:0]	Digital outputs	PLAY internal outputs
play_clk	Clock input	PLAY clock from RCC

Table 201. PLAY internal signals (continued)

Signal name	Signal type	Description
pclk	Clock input	APB bus clock
play_irq	Interrupt request	Global interrupt
play_rstn	Reset input	Global reset (includes config)
play_app_rstn	Reset input	Application reset (excludes config)
play_priv0, play_priv1	Digital outputs	Privilege attribute indicators

Table 202 lists the signals connected to the PLAY1 inputs. Up to 16 inputs can be selected for connection to the logic array, one from each row. See *Section 23.3.4: Input selection* for details.

Table 202. PLAY1 logic array input *n* signal selection(1)

input *n*	PLAY_FILT*n*CFG.PREMUXSEL[2:0]							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	**play_in0_0**	**play_in0_1**	**play_in0_2**	**play_in0_3**	**play_in0_4**	**play_in0_5**	**play_in0_6**	**play_in0_7**
	PLAY_IN0	tim2_trgo	tim3_trgo	hrtim1_dac_reset_trg1	EVENTOUT	-	comp1_out	tim9_trgo
1	**play_in1_0**	**play_in1_1**	**play_in1_2**	**play_in1_3**	**play_in1_4**	**play_in1_5**	**play_in1_6**	**play_in1_7**
	PLAY_IN1	tim2_oc3	tim3_oc3	-	SPI2_NSS	hrtim1_adc_trg1	comp2_out	comp1_out
2	**play_in2_0**	**play_in2_1**	**play_in2_2**	**play_in2_3**	**play_in2_4**	**play_in2_5**	**play_in2_6**	**play_in2_7**
	PLAY_IN2	tim2_oc4	tim3_oc4	tim20_trgo	SPI2_MOSI	hrtim1_adc_trg2	-	-
3	**play_in3_0**	**play_in3_1**	**play_in3_2**	**play_in3_3**	**play_in3_4**	**play_in3_5**	**play_in3_6**	**play_in3_7**
	PLAY_IN3	lptim1_ch1	comp3_out	tim20_oc5	SPI2_SCLK	-	-	-
4	**play_in4_0**	**play_in4_1**	**play_in4_2**	**play_in4_3**	**play_in4_4**	**play_in4_5**	**play_in4_6**	**play_in4_7**
	PLAY_IN4	lptim1_ch2	hrtim1_dac_step_trg1	tim20_oc6	tim3_trgo	-	MCO2	-
5	**play_in5_0**	**play_in5_1**	**play_in5_2**	**play_in5_3**	**play_in5_4**	**play_in5_5**	**play_in5_6**	**play_in5_7**
	PLAY_IN5	comp4_out	comp3_out	hrtim1_out_sync2	tim3_oc3	SPI1_NSS	-	-
6	**play_in6_0**	**play_in6_1**	**play_in6_2**	**play_in6_3**	**play_in6_4**	**play_in6_5**	**play_in6_6**	**play_in6_7**
	PLAY_IN6	MCO1	tim1_oc1	tim15_oc2	tim3_oc4	SPI1_MOSI	-	-
7	**play_in7_0**	**play_in7_1**	**play_in7_2**	**play_in7_3**	**play_in7_4**	**play_in7_5**	**play_in7_6**	**play_in7_7**
	PLAY_IN7	adc1_awd1	adc2_awd1	comp5_out	tim4_trgo	SPI1_SCLK	tim6_trgo	-
8	**play_in8_0**	**play_in8_1**	**play_in8_2**	**play_in8_3**	**play_in8_4**	**play_in8_5**	**play_in8_6**	**play_in8_7**
	PLAY_IN8	adc1_awd2	-	tim20_oc1	tim4_oc4	tim1_trgo	-	-
9	**play_in9_0**	**play_in9_1**	**play_in9_2**	**play_in9_3**	**play_in9_4**	**play_in9_5**	**play_in9_6**	**play_in9_7**
	PLAY_IN9	USART1_TX	-	adc1_awd1	comp6_out	tim1_oc5	SPI2_NSS	-

Table 202. PLAY1 logic array input *n* signal selection(1) (continued)

input *n*	PLAY_FILT*n*CFG.PREMUXSEL[2:0]							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
10	**play_in10_0**	**play_in10_1**	**play_in10_2**	**play_in10_3**	**play_in10_4**	**play_in10_5**	**play_in10_6**	**play_in10_7**
	PLAY_IN10	USART1_CK	-	tim12_trgo	tim8_oc1	tim1_oc6	SPI2_MOSI	HRTIM1_CHA1
11	**play_in11_0**	**play_in11_1**	**play_in11_2**	**play_in11_3**	**play_in11_4**	**play_in11_5**	**play_in11_6**	**play_in11_7**
	PLAY_IN11	NMI	comp7_out	USART1_TX	-	lptim1_ch1	SPI2_SCLK	HRTIM1_CHB1
12	**play_in12_0**	**play_in12_1**	**play_in12_2**	**play_in12_3**	**play_in12_4**	**play_in12_5**	**play_in12_6**	**play_in12_7**
	PLAY_IN12	LOCKUP	-	USART1_CK	tim10_oc1	lptim1_ch2	tim8_trgo	HRTIM1_CHC1
13	**play_in13_0**	**play_in13_1**	**play_in13_2**	**play_in13_3**	**play_in13_4**	**play_in13_5**	**play_in13_6**	**play_in13_7**
	PLAY_IN13	SPI1_NSS	-	NMI2	USART2_TX	comp8_out	tim8_oc5	HRTIM1_CHD1
14	**play_in14_0**	**play_in14_1**	**play_in14_2**	**play_in14_3**	**play_in14_4**	**play_in14_5**	**play_in14_6**	**play_in14_7**
	PLAY_IN14	SPI1_MOSI	-	-	usart2_ck	-	tim8_oc6	HRTIM1_CHE1
15	**play_in15_0**	**play_in15_1**	**play_in15_2**	**play_in15_3**	**play_in15_4**	**play_in15_5**	**play_in15_6**	**play_in15_7**
	PLAY_IN15	SPI1_SCLK	-	LOCKUP2	comp2_out	-	tim15_trgo	HRTIM1_CHF1

1. NMI2, LOCKUP2 signals are equivalent to NMI, LOCKUP.

   NMI2, LOCKUP2 signals are reserved for CPU2 in dual core STM32G5 MCUs.

Table 203. PLAY1 output connections

Signal name	External pins	Internal signals
play_out0	PLAY_OUT0	tim2_ti4_in3 tim1_ti1_in9 tim1_ocref_clr12 tim1_brk_cmp13 dac1_trg_async13 hrtim1_eev1_5 hrtim1_ext_trg1 tim8_brk_cmp13 tim20_brk_cmp13
play_out1	PLAY_OUT1	tim14_ti1_in9 tim1_ti3_in3 tim9_ti2_in9 tim1_brk_cmp14 dac2_trg_async13 hrtim1_eev2_5 hrtim1_ext_trg2 tim20_ocref_clr13 tim4_ocref_clr13
play_out2	PLAY_OUT2	tim2_etr24 tim1_ti2_in9 tim3_ocref_clr13 tim1_brk2_cmp13 dac5_trg_async13 hrtim1_eev3_5 hrtim1_ext_trg3 tim8_brk2_cmp13 tim20_brk2_cmp13
play_out3	PLAY_OUT3	tim2_ti3_in3 tim20_ti1_in9 tim3_ocref_clr12 tim1_brk2_cmp14 dac6_trg_async13 hrtim1_eev4_5 hrtim1_ext_trg4 gpdma1_trig37 tim4_ocref_clr12

Table 203. PLAY1 output connections (continued)

Signal name	External pins	Internal signals
play_out4	PLAY_OUT4	tim3_etr23 tim2_etr23 tim15_ti1_in9 tim15_brk_cmp13 dac7_trg_async13 hrtim1_eev5_5 hrtim1_ext_trg5 tim16_brk_cmp13 tim17_brk_cmp13
play_out5	PLAY_OUT5	tim16_ti1_in9 tim8_ti4_in3 tim15_ti2_in9 tim15_brk_cmp14 dac8_trg_async13 hrtim1_eev6_5 hrtim1_ext_trg6 tim8_ocref_clr12 tim4_etr23
play_out6	PLAY_OUT6	tim1_etr23 tim20_etr23 tim1_ocref_clr13 tim8_etr23 dac9_trg_async13 hrtim1_eev7_5 hrtim1_ext_trg7 comp1_blk15 tim4_ti1_in9
play_out7	PLAY_OUT7	tim12_ti1_in9 tim2_ocref_clr13 tim20_ti2_in9 tim16_brk_cmp14 dac10_trg_async13 hrtim1_eev8_5 hrtim1_ext_trg8 comp2_blk15 tim15_ocref_clr12 posenc1_strobe[17]

Table 203. PLAY1 output connections (continued)

Signal name	External pins	Internal signals
play_out8	PLAY_OUT8	tim1_ti4_in3 tim3_ti1_in9 tim20_ti3_in3 tim1_etr24 dac1_trg_asyncstep19 hrtim1_eev9_5 lptim1_ic1_mux4 comp3_blk15 tim15_ocref_clr13
play_out9	PLAY_OUT9	tim17_ti1_in9 tim3_ti2_in9 tim20_ti4_in3 tim20_brk_cmp14 dac2_trg_asyncstep19 hrtim1_eev10_5 lptim1_ic2_mux4 comp4_blk15 tim16_ocref_clr12
play_out10	PLAY_OUT10	tim10_ti1_in9 tim3_ti3_in3 tim8_ti2_in9 tim8_etr24 dac5_trg_asyncstep19 hrtim1_eev11_5 lptim1_ext_trig7 comp5_blk15 tim16_ocref_clr13 posenc2_strobe[17]
play_out11	PLAY_OUT11	tim11_ti1_in9 tim3_ti4_in3 tim17_ocref_clr12 tim20_brk2_cmp14 dac6_trg_asyncstep19 hrtim1_eev12_5 adc12_ext_trg21 comp6_blk15 tim4_ti2_in9

Table 203. PLAY1 output connections (continued)

Signal name	External pins	Internal signals
play_out12	PLAY_OUT12	tim13_ti1_in9 tim8_ti3_in3 tim20_ocref_clr12 tim17_brk_cmp14 dac7_trg_asyncstep19 hrtim1_eev13_5 adc3_ext_trg21 comp7_blk15 tim4_ti3_in3
play_out13	PLAY_OUT13	tim3_etr24 tim9_ti1_in9 tim17_ocref_clr13 tim8_brk_cmp14 dac8_trg_asyncstep19 hrtim1_eev14_5 adc12_jext_trg21 comp8_blk15 hrtim1_bm_trg
play_out14	PLAY_OUT14	tim20_etr24 tim8_ti1_in9 tim12_ti2_in9 tim8_ocref_clr13 dac9_trg_asyncstep19 hrtim1_eev15_5 adc3_jext_trg21 ext_it51 hrtim1_upd_en1
play_out15	PLAY_OUT15	tim2_ocref_clr12 tim2_ti2_in9 tim2_ti1_in9 tim8_brk2_cmp14 dac10_trg_asyncstep19 hrtim1_eev16_5 tim4_etr24 gpdma2_trig37 hrtim1_in_sync3

23.3.3 PLAY reset and clocks

The PLAY has two reset inputs:

- play_rstn: connected to the power reset
- play_app_rstn: connected to the system reset

The PLAY configuration registers are reset by:

- play_rstn

All other registers, including the PLAY logic element output registers, are reset by:

- play_rstn
- play_app_rstn

The PLAY configuration and control registers are clocked by:

- pclk (APB bus clock)

The synchronisation and filters are clocked by:

- play_clk

The PLAY logic element output registers are clocked by:

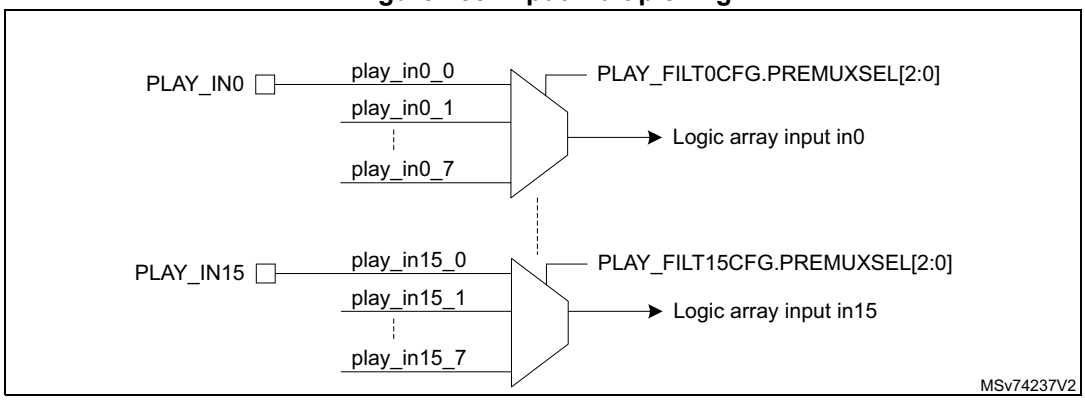
- play_clk, optionally conditioned by any of the PLAY inputs or logic element registered outputs

23.3.4 Input selection

There are 16 inputs to the logic array, in0 to in15, which can be connected to external I/Os or internal signals coming from other functional units.

Each logic array input can connect to one of eight signals. A pre-multiplexer selects which signal is connected to an input. The pre-multiplexer for logic array input *n* is configured by the PLAY_FILT*n*CFG.PREMUXSEL[2:0] register field (see *Figure 155*). Only one signal can be chosen per input.

Figure 155. Input multiplexing

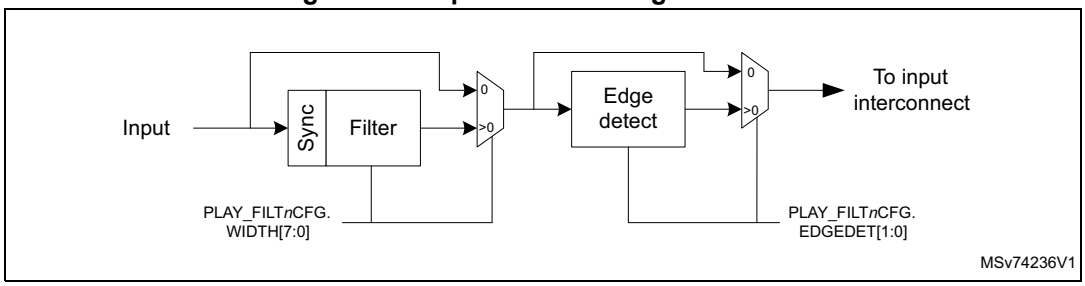


The mapping of signals to logic array inputs is shown in *Table 202*. The left hand column contains the logic array input number. The columns to the right show the play_in signal name as well as the source signal to which it is connected, for each setting of PLAY_FILT*n*CFG.PREMUXSEL[2:0]. Some source signal names appear more than once in the table, which means they can be mapped to more than one input. This offers more flexibility when mapping source signals to logic array inputs.

23.3.5 Input filtering and edge detection

Inputs pass through a filter stage, which can be configured to remove short pulses or glitches and generate a pulse if an edge is detected (see *Figure 156*).

Figure 156. Input filter and edge detection



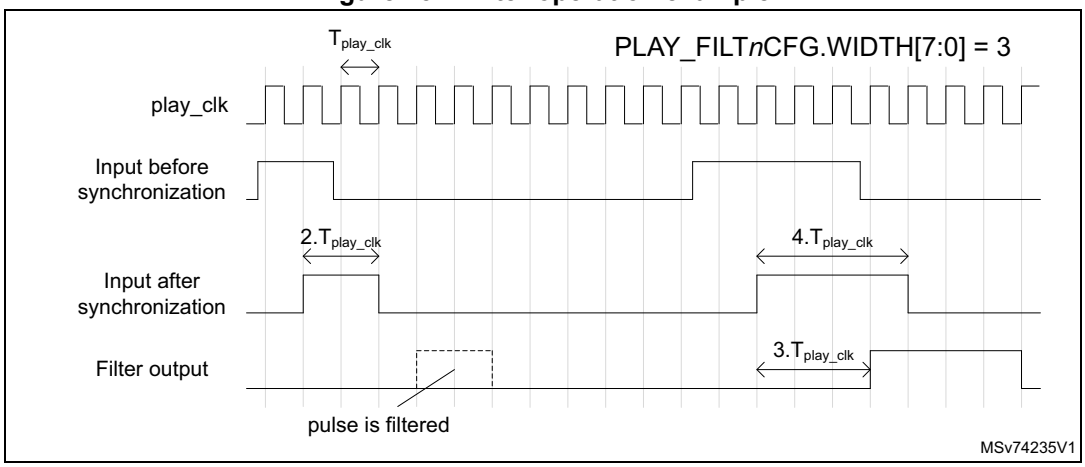
Filtering

Every input has a filter which removes pulses shorter than a programmable number of play_clk clock cycles (after synchronization). The filter samples the input at the play_clk clock frequency and starts a counter if it detects a change in state. If the input transitions again before the counter reaches zero, the counter is stopped and reloaded, and the filter output state remains unchanged. If the counter reaches zero, it is reloaded and the filter output state toggles.

The minimum pulse width for input *n* can be programmed from 1 to 255 cycles in register bits PLAY_FILTCFG*n*.WIDTH[7:0]. Note that the filter introduces a delay equal to the programmed minimum pulse width.

Figure 157 illustrates the operation of the filter for PLAY_FILTCFG*n*.WIDTH[7:0] = 3. Pulses whose width is less than three periods of play_clk (T_{play_clk}) are filtered. Pulses whose width is three periods or more propagate through the filter, with a delay of three T_{play_clk} periods.

Figure 157. Filter operation example



The filter can be bypassed by setting PLAY_FILTnCFG.WIDTH[7:0] = 0.

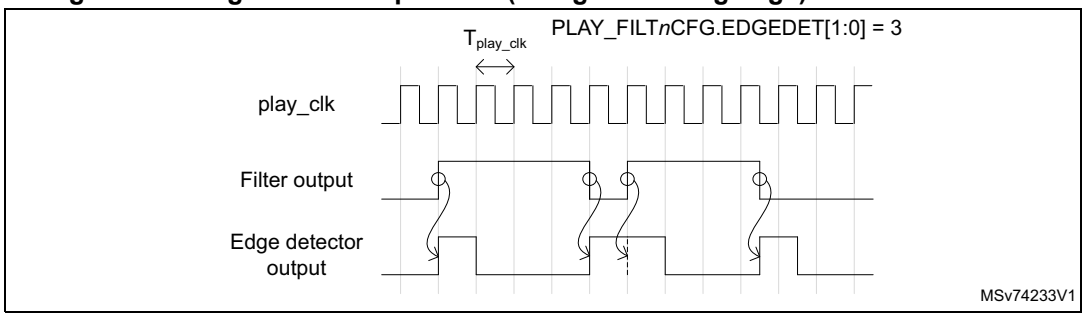
Edge detection and pulse generation

For all inputs there is an optional edge detection feature which detects a transition and generates a positive pulse of one play_clk clock period. This can be used to detect and lengthen pulses which are shorter than the play_clk clock period, or to trigger actions on an edge rather than a level. The edge detection feature is enabled for input *n* by programming

the PLAY_FILTCFG*n*.EDGEDET[1:0] field with direction of the transition (rising, falling or both edges). Setting the field to 0 disables and bypasses the edge detection.

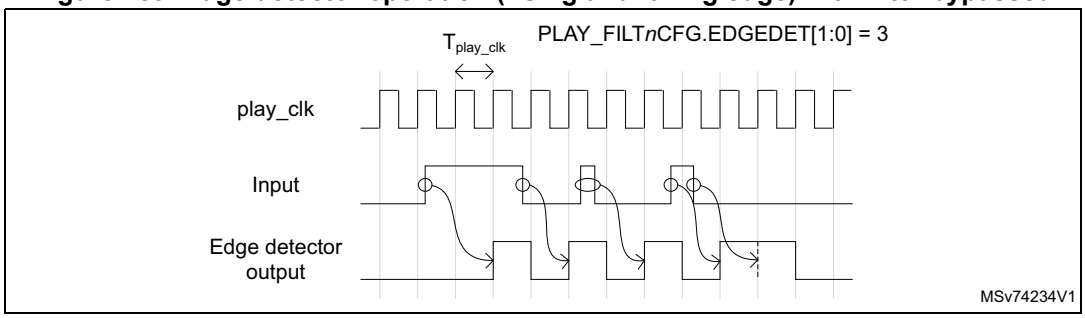
If the filter is enabled (PLAY_FILTCFG*n*.WIDTH[7:0] > 0), the edge detector generates a pulse when an edge is detected at the output of the filter (see *Figure 158*). Note that if both rising and falling edges are programmed, back-to-back transitions on successive clocks generate a double length pulse.

Figure 158. Edge detector operation (rising and falling edge) with filter enabled



To lengthen a pulse shorter than one play_clk period, the filter must be bypassed. This is done by setting PLAY_FILTCFG*n*.WIDTH[7:0] = 0. The synchronization is done by the edge detection unit. When an edge is detected on the input signal, a positive pulse is generated after a delay of between one and two play_clk periods (see *Figure 159*).

Figure 159. Edge detector operation (rising and falling edge) with filter bypassed



If both rising and falling edges are programmed, and the input pulse width is less than one play_clk period, only one output pulse is generated. Note however that if two transitions occur either side of a clock edge, a double length pulse is generated.

Synchronization

All input signals are considered to be asynchronous to the play_clk clock. If any synchronous logic is connected downstream, it is mandatory to synchronize the inputs to the play_clk clock, to avoid meta-stability. Synchronization introduces a delay of 1 to 2 play_clk periods.

Synchronization is performed by enabling the filter or the edge detector. If both the filter and edge detector are bypassed (PLAY_FILT*n*CFG.WIDTH[7:0] = 0 and PLAY_FILT*n*CFG.EDGEDET[1:0] = 0), the input signal is propagated directly to the logic array without synchronization.

Caution: If both edge detector and filter are bypassed, the input must only be used to drive purely combinatorial logic.

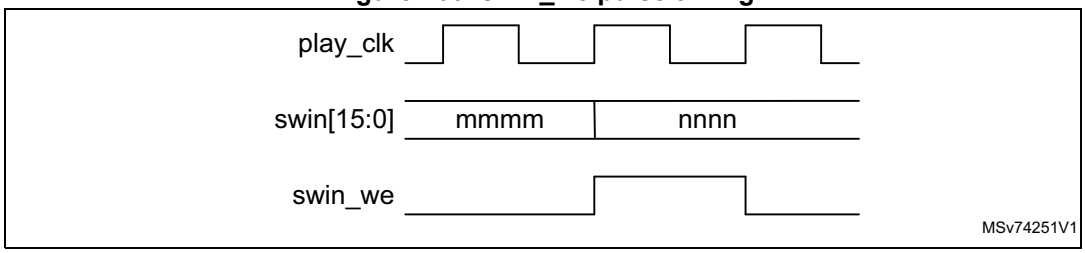
23.3.6 Software programmable inputs

The PLAY_SWIN register contains 16 bits which are connected to inputs swin0 to swin15. This allows software programmable functionality to be included in the logic expression. This can be used for example to enable or disable functionality, or to trigger state transitions, by software. These inputs also allow testing of the programmed logic array functionality.

The PLAY_SWIN register bits can be set independently by writing a 1 to the corresponding bits in the PLAY_SWIN_SET register. In the same way they can be reset by writing a 1 to the corresponding bits in the PLAY_SWIN_CLR register. It is also possible to read and write the bits directly in the PLAY_SWIN register.

Every time there is a write to the PLAY_SWIN, PLAY_SWIN_SET or PLAY_SWIN_CLR register, a positive pulse is generated on the swin_we signal. The pulse lasts one play_clk cycle, starting from the moment the register value is updated (see *Figure 160*). The swin_we signal is connected to the input interconnect and can be used as a clock enable for any of the logic element output registers. This means that writing to the PLAY_SWIN register can trigger a state transition conditioned by the new values on swin0 to swin15, for example.

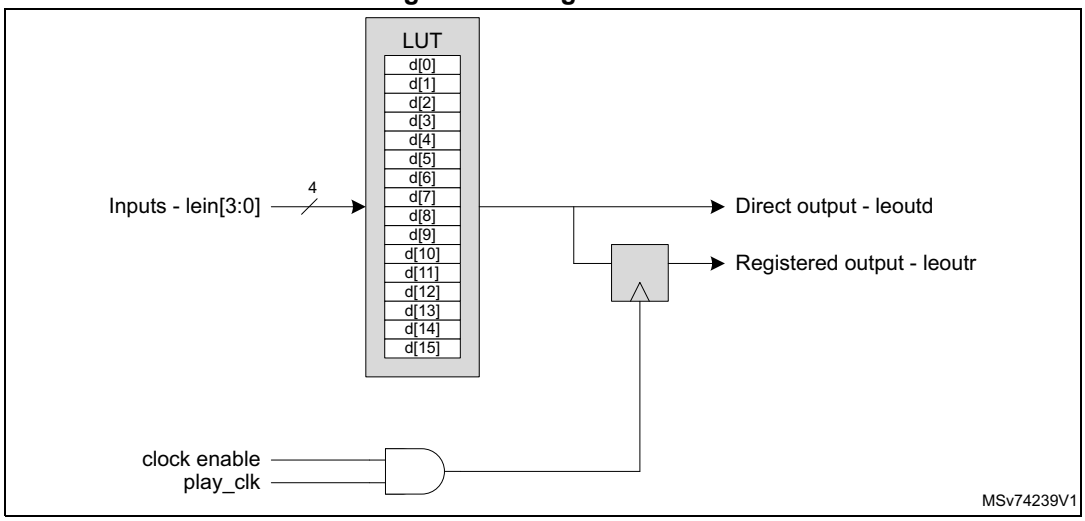
Figure 160. swin_we pulse timing



23.3.7 Logic elements

The logic array provides 16 logic elements (LEs), each containing a 4-input, single output look-up table (LUT) and a register (*Figure 161*).

Figure 161. Logic element



Look-up table (LUT)

The LUT is a 16 x 1-bit memory which generates a single output, leoutd, whose value is a function of the four LUT inputs, lein[3:0] which make up the address. The LUT is programmed at configuration time with the required state of the output for every possible combination of lein[3:0].

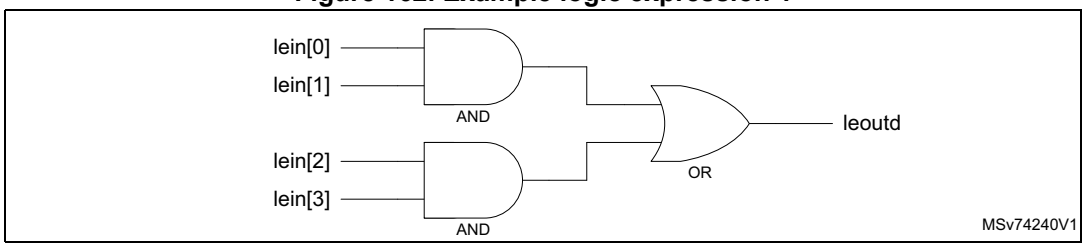
To implement a Boolean logic expression, the full truth table for the expression must be detailed.

This is demonstrated using the following logic expression as an example:

leoutd = lein[0] AND lein[1] OR lein[2] AND lein[3]

This expression corresponds to the diagram in *Figure 162*.

Figure 162. Example logic expression 1



The truth table is given in *Table 204*.

Table 204. Truth table example 1

Inputs - lein[3:0]	Output - leoutd
0000	0
0001	0
0010	0
0011	1
0100	0

Table 204. Truth table example 1 (continued)

Inputs - lein[3:0]	Output - leoutd
0101	0
0110	0
0111	1
1000	0
1001	0
1010	0
1011	1
1100	1
1101	1
1110	1
1111	1

The left hand column lists the address formed by the combination of the four LUT inputs, lein[3:0], and the right hand column lists the required output value on leoutd for each input combination.

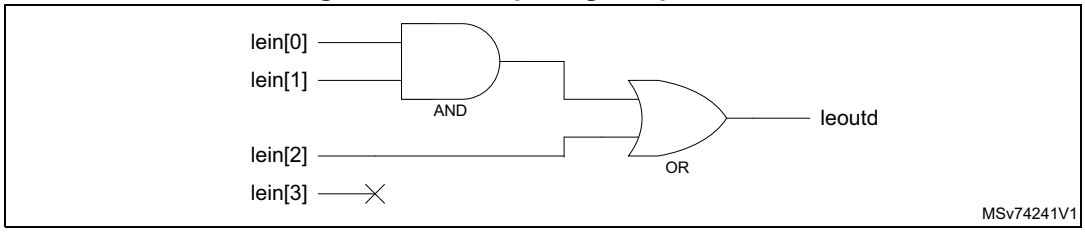
During configuration, the LUT is programmed as a single 16-bit register, with the data read from the output column of the truth table, read from bottom (MSB) to top (LSB). In this example the LUT configuration register would be programmed with 1111 1000 1000 1000 (0xF888).

If any of the inputs are unused or “don’t care”, the output must be the same for both possible values of the unused input. In other words, the contents of address 0bxnnn must be the same for x = 0 or x = 1.

For example, in the following logic expression, lein[3] is unused:

leoutd = lein[0] AND lein[1] OR lein[2]

Figure 163. Example logic expression 2



The truth table for this expression is shown in *Table 205*. To obtain the full LUT, each row containing an ‘x’ is split into two identical rows. Then the ‘x’ is replaced by ‘0’ and ‘1’ respectively.

In this example the configuration register is programmed with 1111 1000 1111 1000 (0xF8F8).

Table 205. Truth table example 2

lein[3:0]	leoutd
x000	0
x001	0
x010	0
x011	1
x100	1
x101	1
x110	1
x111	1

Alternatively, the unused input can be connected to an unused input with value 0 (see *Table 206*), so that addresses with x = 1 are never accessed and can be left at 0.

The LUT configuration registers are directly mapped in the PLAY APB address space. The LUT configuration for logic element *n* is programmed in the register bit field PLAY_LE*n*CFG1.LUT[15:0].

LE output register

The LUT output can be direct or registered. Output registers are clocked by the play_clk, optionally qualified by a clock enable signal. When the clock enable signal is high, the register is updated with the LUT output every play_clk cycle. A clock enable must be high for at least one play_clk cycle. This constraint is guaranteed for input signals using edge detection or filtering (see *Section 23.3.5: Input filtering and edge detection*). It is also guaranteed for the synchronous feedback signals leoutr[15:0].

Caution: Asynchronous signals must not be selected as clock enables.

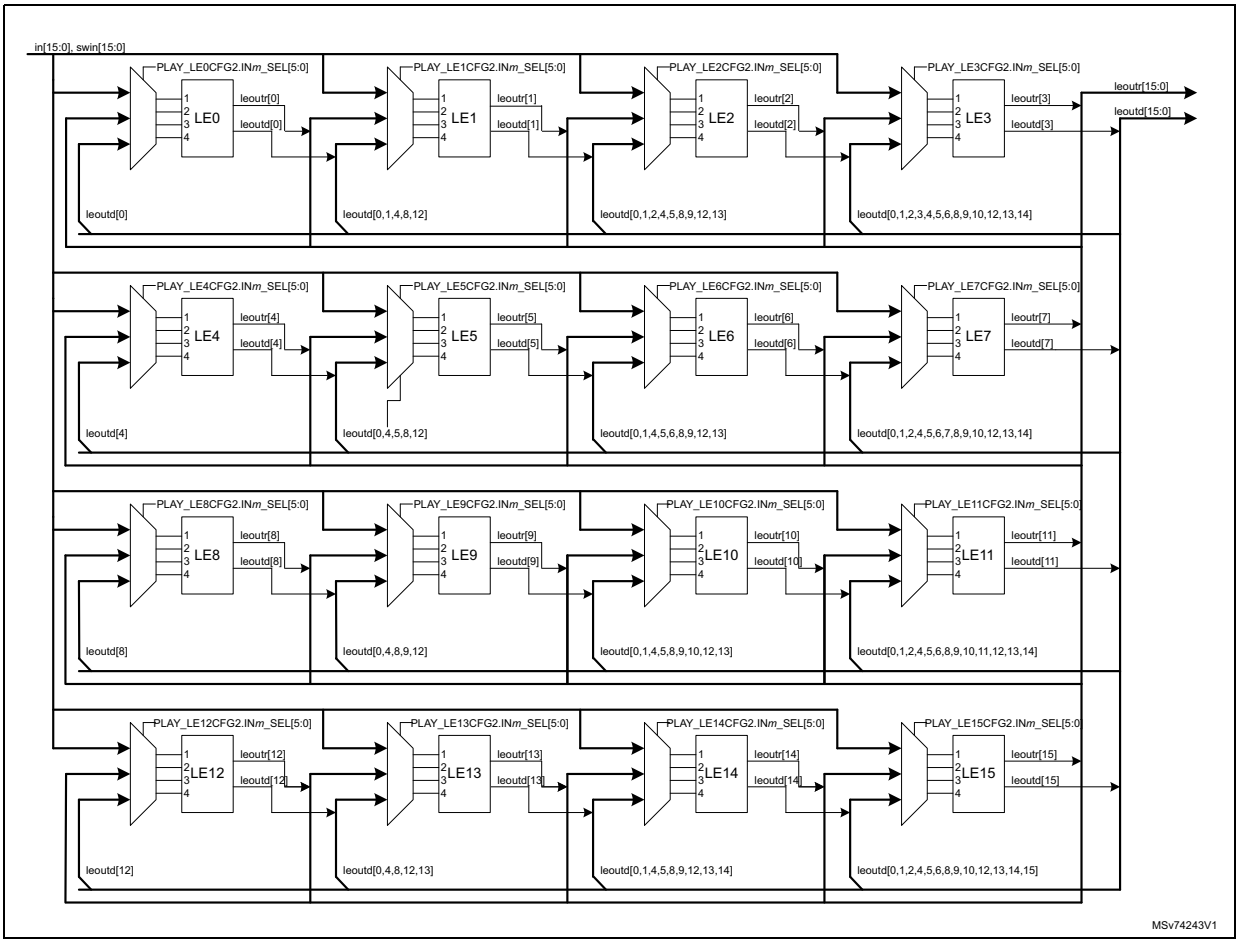
During configuration (when PLAY_CFGCR.UNLOCK = 1), the logic element registers are held in reset.

Logic element registers are also reset by play_rstn and play_app_rstn.

23.3.8 Input interconnect

The input interconnect selects which signals are connected to each logic element, as shown in *Figure 164*.

Figure 164. Input interconnect



LUT input selection

For each logic element, up to four input signals can be configured as LUT inputs, and one as a clock enable for the LE register. The LUT inputs can be selected from among the following signals:

- in[15:0] - 16 inputs from I/Os or other functional blocks, via the filter and edge detection unit
- swin[15:0] - 16 register bits (PLAY_SWIN register)
- leoutr[15:0] - 16 feedback signals from LE registered outputs
- logic 0

Each logic element's direct output can be fed back to one of its inputs. This allows implementation of asynchronous latches.

It is also possible to connect the direct outputs of certain logic elements to the inputs of others - see *Combining logic elements*.

For logic element *n* (0 to 15), the signals connected to input *m* (0 to 3), are selected by configuration register PLAY_LE*n*CFG2, bit field IN*m*_SEL[5:0].

The selection of the LUT input signals is summarized in *Table 206*.

For example, to connect input signal in15 to input 2 of logic element 0, program PLAY_LE0CFG2.IN2_SEL[5:0] = 47. To connect input 3 of logic element 1 to the direct output of logic element 0, leoutd0, program PLAY_LE1CFG2.IN3_SEL[5:0] = 0.

Table 206. Logic element *n* LUT input selection

PLAY_LE*n*CFG2.IN*m*_SEL [5:0]	Signal connected to LUT input *m*	Source
0	For *n* = 4,8,12: 0	-
	For *n* = 0,1,2,3,5,6,7,9,10,11,13,14,15: leoutd[*0*]	LE 0 direct output
1	For *n* = 0,4,5,8,9,12,13: 0	-
	For *n* = 1,2,3,6,7,10,11,14,15: leoutd[*1*]	LE 1 direct output
2	For *n* = 0,1,4,5,6,8,9,10,12,13,14: 0	-
	For *n* = 2,3,7,11,15: leoutd[*2*]	LE 2 direct output
3	For *n* = 0,1,2,4,5,6,7,8,9,10,11,12,13,14,15: 0	-
	For *n* = 3: leoutd[*3*]	LE 3 direct output
4	For *n* = 0,8,12: 0	-
	For *n* = 1,2,3,4,5,6,7,9,10,11,13,14,15: leoutd[*4*]	LE 4 direct output
5	For *n* = 0,1,4,8,9,12,13: 0	-
	For *n* = 2,3,5,6,7,10,11,14,15: leoutd[*5*]	LE 5 direct output
6	For *n* = 0,1,2,4,5,8,9,10,12,13,14: 0	-
	For *n* = 3,6,7,11,15: leoutd[*6*]	LE 6 direct output
7	For *n* = 0,1,2,3,4,5,6,8,9,10,11,12,13,14,15: ’0’	-
	For *n* = 7: leoutd[*7*]	LE 7 direct output
8	For *n* = 0,4,12: 0	-
	For *n* = 1,2,3,5,6,7,8,9,10,11,13,14,15: leoutd[*8*]	LE 8 direct output
9	For *n* = 0,1,4,5,8,12,13: 0	-
	For *n* = 2,3,6,7,9,10,11,14,15: leoutd[*9*]	LE 9 direct output
10	For *n* = 0,1,2,4,5,6,8,9,12,13,14,15: 0	-
	For *n* = 3,7,10,11: leoutd[*10*]	LE 10 direct output
11	For *n* = 0,1,2,3,4,5,6,7,8,9,10,12,13,14,15: 0	-
	For *n* = 11: leoutd[*11*]	LE 11 direct output
12	For *n* = 0,4,8: 0	-
	For *n* = 1,2,3,5,6,7,9,10,11,12,13,14,15: leoutd[*12*]	LE 12 direct output
13	For *n* = 0,1,4,5,8,9,12: 0	-
	For *n* = 2,3,6,7,10,11,13,14,15: leoutd[*13*]	LE 13 direct output

Table 206. Logic element *n* LUT input selection (continued)

PLAY_LE*n*CFG2.IN*m*_SEL [5:0]	Signal connected to LUT input *m*	Source
14	For *n* = 0,1,2,4,5,6,8,9,10,12,13: 0	-
	For *n* = 3,7,11,14,15: leoutd[*14*]	LE 14 direct output
15	For *n* = 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14: 0	-
	For *n* = 15: leoutd[*15*]	LE 15 direct output
16 to 31	leoutr[0] to leoutr[15]	LE registered outputs
32 to 47	in0 to in15	input signals
48 to 63	swin0 to swin15	PLAY_SWIN

LE clock enable selection

The LE register clock enable can be selected from any of these signals:

- in[15:0] - 16 inputs from I/Os or other functional blocks, via the filter and edge detection
- swin[15:0] - 16 register bits (PLAY_SWIN register)
- swin_we - pulse indicating that a write to the PLAY_SWIN register has taken place
- leoutr[15:0] - 16 feedback signals from LE registered outputs
- 0- clock disabled
- 1- always enabled

For logic element n the clock enable is selected by configuration register bits PLAY_LE*n*CFG2.CK_SEL[5:0].

The selection of the LE register clock is summarized in *Table 207*.

Table 207. Logic element *n* register clock selection

PLAY_LE*n*CFG2.CK_SEL[5:0]	Clock enable signal[(1)]	Source
0	0 (clock disabled)	-
1	1 (always enabled)	-
2	swin_we	APB interface
3 to 15	reserved	-
16 to 31	leoutr0 to leoutr15 (excluding leoutr*n*)	LE registered outputs
32 to 47	in0 to in15	input signals
48 to 63	swin0 to swin15	PLAY_SWIN

1. Reserved values select 0

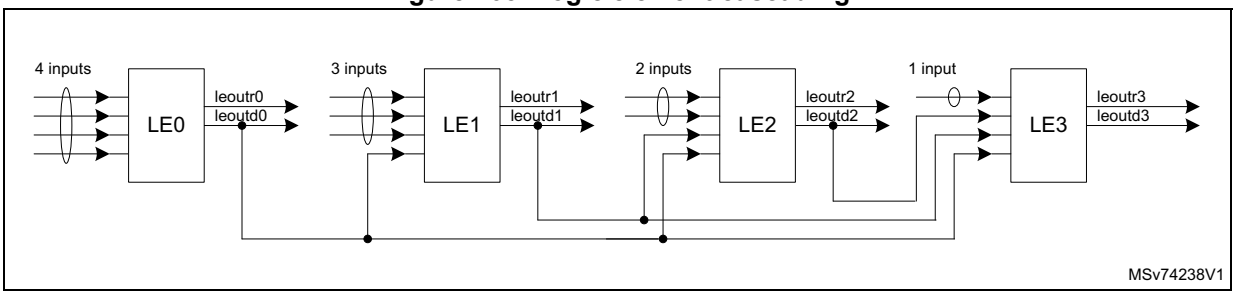
Combining logic elements

LE 0-3 form a group which can be cascaded combinatorially from left to right (see *Figure 165*):

- LE0 direct output can be connected to LE1-3 inputs
- LE1 direct output can be connected to LE2-3 inputs
- LE2 direct output can be connected to LE3 inputs

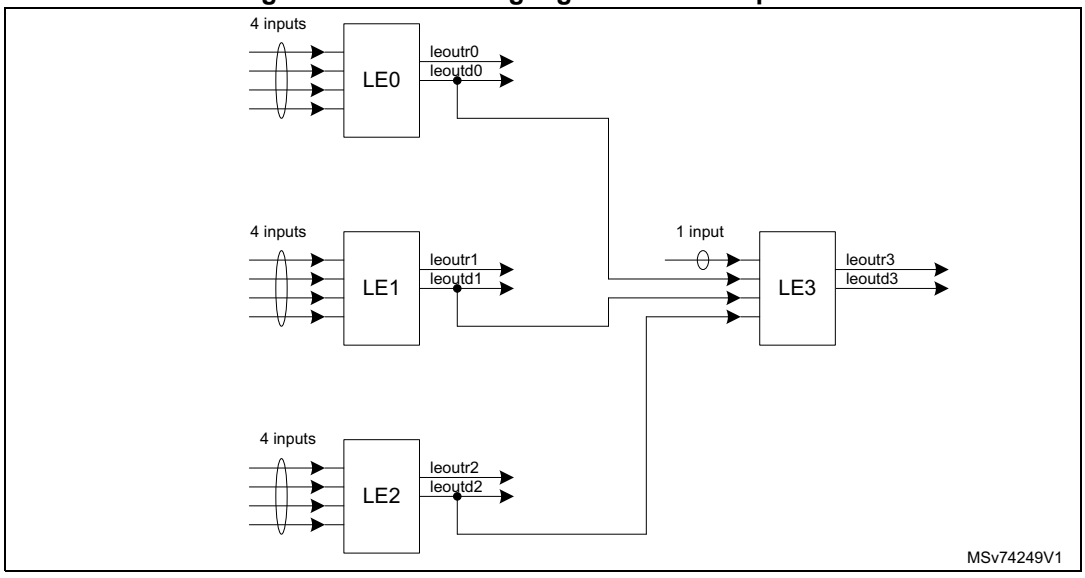
The other groups of four (LE 4-7, LE 8-11, LE12-15) can be cascaded in the same way. Furthermore, logic elements from different groups can be connected together, provided the left to right logic flow is respected. For example, LE4 direct output can be connected to LE1 input, but not to LE0 input.

Figure 165. Logic element cascading



It is also possible to use one logic element to combine the outputs of up to three others, as shown in *Figure 166*. In this example, the direct outputs of LE0-2 are connected to the inputs of LE3. This allows up to 13 different inputs to be combined in a single logic expression.

Figure 166. Combining logic elements in parallel



Note that combining logic elements in this way requires the logic expression to be split into partial expressions, with up to four inputs each.

For example, consider the following expression:

Y = (X1 AND X2) OR (X3 AND X4) OR (X5 AND X6).

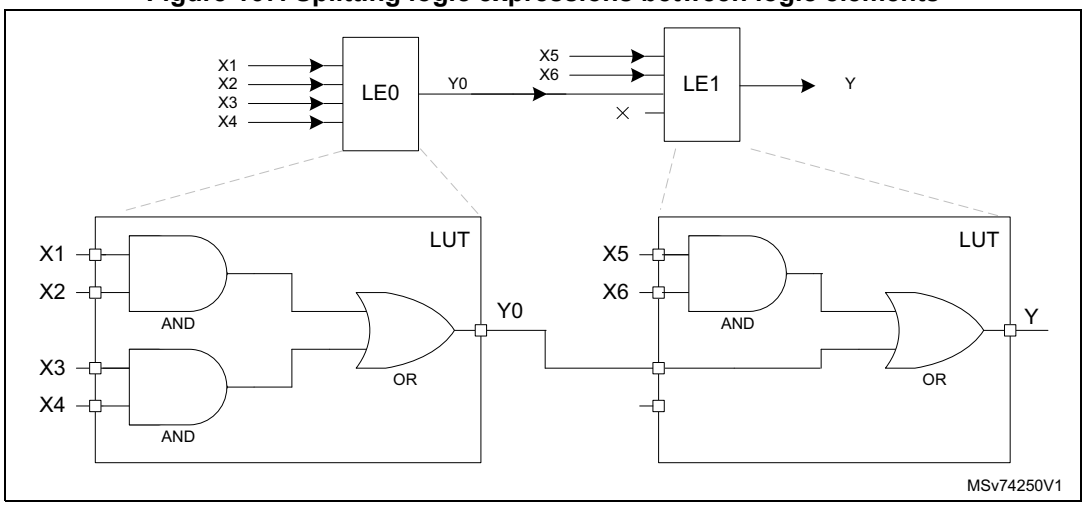
This expression has six inputs (X1 to X6) and one output (Y). It therefore requires at least two logic elements, for example LE0 and LE1, as illustrated in *Figure 167*. The expression for Y can be split into two expressions:

a) Y0 = (X1 AND X2) OR (X3 AND X4)

b) Y = Y0 OR (X5 AND X6)

Then LE0 is configured with expression ‘a’, and LE1 with expression ‘b’. Inputs X1 to X4 are connected to LE0. Inputs X5 and X6, are connected to LE1, together with Y0, the direct output of LE0.

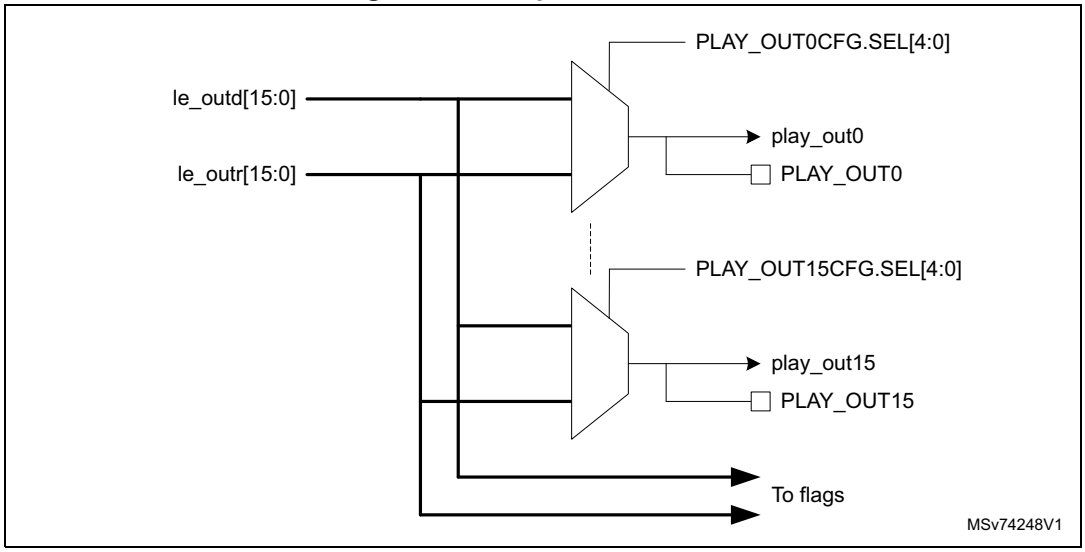
Figure 167. Splitting logic expressions between logic elements



23.3.9 Output interconnect

The output interconnect selects which logic element or eFPGA outputs are connected to which PLAY outputs, as shown in *Figure 168*.

Figure 168. Output interconnect



The PLAY outputs include the following signals:

- play_out[15:0] - 16 outputs to I/Os or other functional blocks

Each output can be connected to any one of the logic element outputs, registered or direct. The signal connected to output *n* is selected by configuration register bits PLAY_OUT*n*CFG.SEL[4:0].

The output selection is summarized in *Table 208*.

Table 208. Output n source selection

PLAY_OUT*n*CFG.SEL[4:0]	Source
0	leoutd0 - LE0 direct output
1	leoutd1 - LE1 direct output
2	leoutd2 - LE2 direct output
3	leoutd3 - LE3 direct output
4	leoutd4 - LE4 direct output
5	leoutd5 - LE5 direct output
6	leoutd6 - LE6 direct output
7	leoutd7 - LE7 direct outpu
8	leoutd8 - LE8 direct output
9	leoutd9 - LE9 direct output
10	leoutd10 - LE10 direct output

Table 208. Output n source selection (continued)

PLAY_OUT*n*CFG.SEL[4:0]	Source
11	leoutd11 - LE11 direct output
12	leoutd12 - LE12 direct output
13	leoutd13 - LE13 direct output
14	leoutd14 - LE14 direct output
15	leoutd15 - LE15 direct output
16	leoutr0 - LE0 register output
17	leoutr1 - LE1 register output
18	leoutr2 - LE2 register output
19	leoutr3 - LE3 register output
20	leoutr4 - LE4 register output
21	leoutr5 - LE5 register output
22	leoutr6 - LE6 register output
23	leoutr7 - LE7 register output
24	leoutr8 - LE8 register output
25	leoutr9 - LE9 register output
26	leoutr10 - LE10 register output
27	leoutr11 - LE11 register output
28	leoutr12 - LE12 register output
29	leoutr13 - LE13 register output
30	leoutr14 - LE14 register output
31	leoutr15 - LE15 register output

23.3.10 Flags

A transition on any of the logic element outputs leoutd[15:0] or leoutr[15:0] can set the corresponding flag in the PLAY_FLSTAT register.

The direction of the transition (rising or falling) can be selected in the PLAY_FLCTL register. Once set, the flag can only be reset by software writing to the PLAY_FLCLR register (or by a reset). Setting a flag generates an interrupt to the CPU on the play_irq signal, if the enable bit for the flag is set in the PLAY_FLIER register. To clear the interrupt the software must reset the flag, or mask it by resetting the enable.

Due to the asynchronous relationship between play_clk and the pclk, a resynchronisation delay is incurred between a transition on the logic element output and the setting of the corresponding flag. The maximum delay is approximately one period of play_clk plus three periods of pclk. The play_irq interrupt is not asserted until the flag goes high. Note that if successive transitions occur on the same output before the flag is cleared, they are not taken into account. However if a subsequent transition on another logic element output occurs during the resynchronization delay, a second resynchronization starts as soon as the first is finished, at the end of which the second flag is set.

The raw state of the signals can also be read by software via the PLAY_OSR register. This is primarily intended for debugging the programmed logic. The user must be aware that the logic element outputs can change asynchronously and the value that is read from PLAY_OSR may not correspond to the actual state of the outputs at the moment of reading.

23.3.11 Miscellaneous status bits

The software input register (PLAY_SWIN), and the flag edge control register (PLAY_FLCTL) are in the play_clk domain. When a write occurs to one of these registers (including to PLAY_SWINSET/CLR), the write data is stored temporarily in the APB bus clock (pclk) domain until the next play_clk pulse at which point the data is transferred to the target register. If play_clk is slow compared to pclk, or if it is inactive, the update could take an indefinitely long time to complete. To avoid stalling the processor, any subsequent writes to the same register before the target register update has completed, are ignored.

To ensure that updates are not lost, the PLAY_MSR register contains two read-only status bits, SWIN write busy flag (SWIN_WBFS) and FLCTL write busy flag (FLCTL_WBFS). When one of these bits is set to 1, it indicates that an update is pending to the corresponding register, PLAY_SWIN or PLAY_FLCTL. The busy flags are automatically reset to 0 when the update completes.

If play_clk is significantly slower than pclk, it is recommended to poll the busy flag in the PLAY_MSR prior to writing to the corresponding register, to be sure that it has reset to 0 indicating that the previous update has completed.

Alternatively, software can enable the write complete interrupts in the PLAY_IER register to generate an interrupt request when a pending write completes (see *Section 23.5*).

23.4 PLAY in low-power modes

The play_clk can be maintained when the system is in low-power mode (Stop mode). The full functionality of the logic array is available, but the memory-mapped registers are not accessible and the flags are not updated.

When the play_clk is stopped, the logic element output registers do not operate, neither does the edge detection and filtering. However combinatorial logic continues to function as programmed.

If a logic element direct output is mapped to an asynchronous wake-up event, a transition on the output can wake the system from Stop mode. It is recommended to configure one of the software programmable inputs to reset the output so that software can deassert the wake-up event, if it is not otherwise deasserted.

In Standby mode, the PLAY is powered down and no longer functional. It must be reconfigured on exit from Standby.

23.5 PLAY interrupts

There is one interrupt request, play_irq, which can be configured to interrupt the processor when any of the following conditions occurs:

1. One or more of the flags are set, as indicated by the PLAY_FLSTAT register
2. An outstanding write to the PLAY_FLCTL register has completed, as indicated by the FLCTLWBFS bit in the PLAY_MSR register.
3. An outstanding write to the PLAY_SWIN register has completed, as indicated by the SWINWBFS bit in the PLAY_MSR register.

An interrupt handler must first read the PLAY interrupt status register (PLAY_ISR) to determine the source of the interrupt.

If the FLCTL write complete (FLCTLWC) or SWIN write complete (SWINWC) bits are set, it means that a previous write respectively to the PLAY_FLCTL or PLAY_SWIN register has completed. The write complete status bits can be cleared by writing a one to the corresponding bit in the interrupt clear (PLAY_ICR) register. The write complete interrupts are enabled and disabled by writing to the PLAY_IER register.

If the PLAY_ISR.FLAGS bit is set, the interrupt handler must read the PLAY_FLSTAT register to determine which flag(s) caused the interrupt. The FLAGS bit is reset by hardware when all the flags are inactive or masked. See *Section 23.3.10* for more details.

23.6 PLAY registers

There are three types of PLAY registers:

- Access control registers:

  PLAY_SECCFGR

  PLAY_PRIVCFGR

  These registers control software access rights to the PLAY registers.
- Configuration registers: PLAY_CFGCR, PLAY_FILTxCFGR, PLAY_LExCFG1, PLAY_LExCFG2, PLAY_OUTxCFG

  These registers configure the logic elements, input and output interconnects, input multiplexers and filters.
- Functional registers: PLAY_SWIN, PLAY_SWINSET, PLAY_SWINCLR, PLAY_OSR, PLAY_FLSTAT, PLAY_FLSET, PLAY_FLCLR, PLAY_FLIER, PLAY_FLCTL, PLAY_MSR, PLAY_ISR, PLAY_IER, PLAY_ICR

  These registers are used by the application software to interact with the logic array during operation.

23.6.1 PLAY configuration control register (PLAY_CFGCR)

Address offset: 0x000

Reset value: 0x0000 0001

This register contains miscellaneous configuration bits. It is a configuration register and is reset by play_rstn.

If PLAY_SECCFGR.SEC[0] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[0] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **UNLOCK**: Unlock configuration

When configuration is complete, it must be set to 0 to prevent accidental corruption of the configuration registers and enable operation of the logic element registers.

0: Configuration is locked. All configuration registers are write protected.

1: Configuration is unlocked. Configuration registers can be programmed.

23.6.2 PLAY security attributes configuration register (PLAY_SECCFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register controls the security attributes of the PLAY registers. It is an access control register and is reset by play_rstn.

The register can only be modified by a secure, privileged write access. It can be read by any access. A nonsecure write access is ignored. An unprivileged write access is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **SEC**[1:0]: Secure access control

0: All registers are accessible in nonsecure or secure state

1: Configuration registers can only be accessed in secure state. All other registers can be accessed in secure or nonsecure state.

2: Reserved

3: All registers are accessible in secure state only.

23.6.3 PLAY privilege attributes configuration register (PLAY_PRIVCFGR)

Address offset: 0x008

Reset value: 0x0000 0000

This register controls the privilege attributes of the PLAY registers. It is an access control register and is reset by play_rstn. It can only be modified by a privileged write access. It can be read by any access.

If PLAY_SECFGR.SEC[n] = 1, PLAY_PRIVCFG.PRIV[n] can only be modified by a secure write access. A nonsecure write access to PLAY_PRIVCFG.PRIV[n] is ignored.

If PLAY_SECCFGR.SEC[n] = 0, PLAY_PRIVCFG.PRIV[n] can be modified by secure or nonsecure write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **PRIV**[1:0]: Privilege access control

0: All registers are accessible in unprivileged or privileged state

1: Configuration registers can only be accessed in privileged state. All other registers can be accessed in privileged or unprivileged state.

2: Reserved

3: All registers are accessible in privileged state only.

23.6.4 PLAY filter x configuration register (PLAY_FILTxCFG)

Address offset: 0x020 + 0x04 * x, (x = 0 to 15)

Reset value: 0x0000 0000

This register configures the filter, edge detector and premultiplexer for logic array input x. It is a configuration register and is reset by play_rstn. Write accesses to this register are ignored if PLAY_CFGCR.UNLOCK = 0.

If PLAY_SECCFGR.SEC[0] = 1, the register requires secure access (nonsecure accesses are ignored, and signaled on play_sec_ilac).

If PLAY_PRIVCFGR.PRIV[0] = 1, the register requires privileged access (unprivileged accesses are ignored, and signaled on play_priv_ilac).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREMUXSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EDGEDET[1:0]		WIDTH[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **PREMUXSEL[2:0]**: Input pre-multiplexer select

Selects the signal to be connected to the input. See *Table 202* for details.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **EDGEDET[1:0]**: Edge detection mode.

0: Edge detection is bypassed

1: A rising edge transition on the input is converted into a positive pulse of 1 play_clk cycle.

2: A falling edge transition on the input is converted into a positive pulse of 1 play_clk cycle.

3: A rising or falling edge transition on the input is converted into a positive pulse of 1 play_clk cycle.

Bits 7:0 **WIDTH[7:0]**: Minimum pulse width

0: Filter is bypassed.

1 to 255: Minimum pulse width, in play_clk cycles. Any pulse on the input shorter than this is ignored.

23.6.5 PLAY logic element x configuration register 1 (PLAY_LExCFG1)

Address offset: 0x060 + 0x04 * x, (x = 0 to 15)

Reset value: 0x0000 0000

This register configures the look-up table contents for logic element x. It is a configuration register and is reset by play_rstn. Write accesses to this register are ignored if PLAY_CFGCR.UNLOCK = 0.

If PLAY_SECCFGR.SEC[0] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[0] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LUT[15:0]**: Look-up table contents.
Truth table output for the logic element.

23.6.6 PLAY logic element x configuration register 2 (PLAY_LExCFG2)

Address offset: 0x0A0 + 0x04 * x, (x = 0 to 15)

Reset value: 0x00FF FFFF

This register configures the input source and the register clock source for logic element x. It is a configuration register and is reset by play_rstn. Write accesses to this register are ignored if PLAY_CFGCR.UNLOCK = 0.

If PLAY_SECCFGR.SEC[0] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[0] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CK_SEL[5:0]						IN3_SEL[5:0]						IN2_SEL[5:4]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IN2_SEL[3:0]				IN1_SEL[5:0]						IN0_SEL[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **CK_SEL[5:0]**: Clock enable selection
Selects source to be used as the clock enable for the logic element register. See *Table 207* for details.
After reset, this bit field is set to 0x0 (disabled)

Bits 23:18 **IN3_SEL[5:0]**: Logic element input 3 selection
Selects source for LUT input 3. See *Table 206* for details.
After reset, this bit field is set to 0x1F: PLAY_SWIN[15]

Bits 17:12 **IN2_SEL[5:0]**: Logic element input 2 selection
Selects source for LUT input 2. See *Table 206* for details.
After reset, this bit field is set to 0x1F: PLAY_SWIN[15]

Bits 11:6 **IN1_SEL[5:0]**: Logic element input 1 selection
Selects source for LUT input 1. See *Table 206* for details.
After reset, this bit field is set to 0x1F: PLAY_SWIN[15]

Bits 5:0 **IN0_SEL[5:0]**: Logic element input 0 selection
Selects source for LUT input 0. See *Table 206* for details.
After reset, this bit field is set to 0x1F: PLAY_SWIN[15]

23.6.7 PLAY output x configuration register (PLAY_OUTxCFG)

Address offset: 0x100 + 0x04 * x, (x = 0 to 15)

Reset value: 0x0000 0000

This register selects the source for play_out[x]. It is a configuration register and is reset by play_rstn. Write accesses to this register are ignored if PLAY_CFGCR.UNLOCK = 0.

If PLAY_SECCFGR.SEC[0] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[0] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEL[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **SEL[4:0]**: Output source selection.
The value of this field selects which LE output is connected to the output. See *Table 208* for details.

23.6.8 PLAY software input status register (PLAY_SWIN)

Address offset: 0x200

Reset value: 0x0000 0000

This register is used to read and write the state of the software programmable inputs. It is a functional register and is reset by play_rstn and play_app_rstn.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored, and signaled on play_sec_ilac).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

Caution: play_clk must be active when writing to this register. Any subsequent write access to this register (or PLAY_SWINSET or PLAY_SWINCLR) is ignored until the write completes. Software must poll the PLAY_SWIN write busy flag in the *PLAY miscellaneous status register (PLAY_MSR)* to make sure that no writes are pending. If play_clk is slow compared to the bus clock, pclk, a write may take a long time to complete. In this case, the PLAY_SWIN register write complete interrupt - see *PLAY interrupt status register (PLAY_ISR)* - can be used to signal to the software when the registers can be accessed again.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIN15	SWIN14	SWIN13	SWIN12	SWIN11	SWIN10	SWIN9	SWIN8	SWIN7	SWIN6	SWIN5	SWIN4	SWIN3	SWIN2	SWIN1	SWIN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SWINy**: Input swiny state (y = 15 to 0)
0: Input low
1: Input high

23.6.9 PLAY software input set register (PLAY_SWINSET)

Address offset: 0x204

Reset value: 0x0000 0000

This register is used to set the state of the software programmable inputs. It is a functional register and is reset automatically by hardware.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

Caution: play_clk must be active when writing to this register. Any subsequent write access to this register (or PLAY_SWIN or PLAY_SWINCLR) is ignored until the write completes. Software must poll the PLAY_SWIN write busy flag in the *PLAY miscellaneous status register (PLAY_MSR)* to make sure that no writes are pending. If play_clk is slow compared to the

bus clock, pclk, a write may take a long time to complete. In this case, the PLAY_SWIN register write complete interrupt - see *PLAY interrupt status register (PLAY_ISR)* - can be used to signal to the software when the register can be accessed again.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIN15	SWIN14	SWIN13	SWIN12	SWIN11	SWIN10	SWIN9	SWIN8	SWIN7	SWIN6	SWIN5	SWIN4	SWIN3	SWIN2	SWIN1	SWIN0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SWINy**: Set PLAY_SWIN.SWINy (y = 15 to 0)
0: Input low
1: Input high

23.6.10 PLAY software input clear register (PLAY_SWINCLR)

Address offset: 0x208

Reset value: 0x0000 0000

This register is used to reset the state of the software programmable inputs. It is a functional register and is reset automatically by hardware.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

Caution: play_clk must be active when writing to this register. Any subsequent write access to this register (or PLAY_SWIN or PLAY_SWINSET) is ignored until the write completes. Software must poll the PLAY_SWIN write busy flag in the *PLAY miscellaneous status register (PLAY_MSR)* to make sure that no writes are pending. If play_clk is slow compared to the bus clock, pclk, a write may take a long time to complete. In this case, the PLAY_SWIN register write complete interrupt - see *PLAY interrupt status register (PLAY_ISR)* - can be used to signal to the software when the registers can be accessed again.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIN15	SWIN14	SWIN13	SWIN12	SWIN11	SWIN10	SWIN9	SWIN8	SWIN7	SWIN6	SWIN5	SWIN4	SWIN3	SWIN2	SWIN1	SWIN0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SWINy**: Reset PLAY_SWIN.SWINy (y = 15 to 0)
0: Input low
1: Input high

23.6.11 PLAY output status register (PLAY_OSR)

Address offset: 0x210

Reset value: 0x0000 0000

This register allows software to read the state of the LE direct and registered outputs. It is a functional register and is reset by play_rstn and play_app_rstn.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LEOUTR15	LEOUTR14	LEOUTR13	LEOUTR12	LEOUTR11	LEOUTR10	LEOUTR9	LEOUTR8	LEOUTR7	LEOUTR6	LEOUTR5	LEOUTR4	LEOUTR3	LEOUTR2	LEOUTR1	LEOUTR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LEOUTD15	LEOUTD14	LEOUTD13	LEOUTD12	LEOUTD11	LEOUTD10	LEOUTD9	LEOUTD8	LEOUTD7	LEOUTD6	LEOUTD5	LEOUTD4	LEOUTD3	LEOUTD2	LEOUTD1	LEOUTD0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **LEOUTRy**: State of LE registered output leoutry (y = 15 to 0)

Bits 15:0 **LEOUTDy**: State of LE direct outputs leoutdy (y = 15 to 0)

23.6.12 PLAY flag status register (PLAY_FLSTAT)

Address offset: 0x220

Reset value: 0x0000 0000

This register allows software to read the state of the LE output flags. It is a functional register and is reset by play_rstn and play_app_rstn.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLAGR15	FLAGR14	FLAGR13	FLAGR12	FLAGR11	FLAGR10	FLAGR9	FLAGR8	FLAGR7	FLAGR6	FLAGR5	FLAGR4	FLAGR3	FLAGR2	FLAGR1	FLAGR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAGD15	FLAGD14	FLAGD13	FLAGD12	FLAGD11	FLAGD10	FLAGD9	FLAGD8	FLAGD7	FLAGD6	FLAGD5	FLAGD4	FLAGD3	FLAGD2	FLAGD1	FLAGD0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **FLAGRy**: Flag state for LEy registered output (y = 15 to 0)

0: Flag reset

1: Flag set

Bits 15:0 **FLAGDy**: Flag state for LEy direct output (y = 15 to 0)
0: Flag reset
1: Flag set

23.6.13 PLAY flag set register (PLAY_FLSET)

Address offset: 0x224

Reset value: 0x0000 0000

This register allows software to set the LE output flags. It is a functional register and is reset automatically by hardware.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLAGR_SET15	FLAGR_SET14	FLAGR_SET13	FLAGR_SET12	FLAGR_SET11	FLAGR_SET10	FLAGR_SET9	FLAGR_SET8	FLAGR_SET7	FLAGR_SET6	FLAGR_SET5	FLAGR_SET4	FLAGR_SET3	FLAGR_SET2	FLAGR_SET1	FLAGR_SET0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAGD_SET15	FLAGD_SET14	FLAGD_SET13	FLAGD_SET12	FLAGD_SET11	FLAGD_SET10	FLAGD_SET9	FLAGD_SET8	FLAGD_SET7	FLAGD_SET6	FLAGD_SET5	FLAGD_SET4	FLAGD_SET3	FLAGD_SET2	FLAGD_SET1	FLAGD_SET0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **FLAGR_SETy**: Flag set for LEy registered output (y = 15 to 0)
0: No effect
1: Set flag

Bits 15:0 **FLAGD_SETy**: Flag set for LEy direct output (y = 15 to 0)
0: No effect
1: Set flag

23.6.14 PLAY flag clear register (PLAY_FLCLR)

Address offset: 0x228

Reset value: 0x0000 0000

This register allows software to clear the state of the LE output flags. It is a functional register and is reset automatically by hardware.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLAGR_CLR15	FLAGR_CLR14	FLAGR_CLR13	FLAGR_CLR12	FLAGR_CLR11	FLAGR_CLR10	FLAGR_CLR9	FLAGR_CLR8	FLAGR_CLR7	FLAGR_CLR6	FLAGR_CLR5	FLAGR_CLR4	FLAGR_CLR3	FLAGR_CLR2	FLAGR_CLR1	FLAGR_CLR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAGD_CLR15	FLAGD_CLR14	FLAGD_CLR13	FLAGD_CLR12	FLAGD_CLR11	FLAGD_CLR10	FLAGD_CLR9	FLAGD_CLR8	FLAGD_CLR7	FLAGD_CLR6	FLAGD_CLR5	FLAGD_CLR4	FLAGD_CLR3	FLAGD_CLR2	FLAGD_CLR1	FLAGD_CLR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **FLAGR_CLRy**: Flag clear for LEy registered output (y = 15 to 0)
0: No effect
1: Clear flag

Bits 15:0 **FLAGD_CLRy**: Flag clear for LEy direct output (y = 15 to 0)
0: No effect
1: Clear flag

23.6.15 PLAY flag interrupt enable register (PLAY_FLIER)

Address offset: 0x22C

Reset value: 0x0000 0000

This register enables interrupt requests based on LE output flag state. It is a functional register and is reset by play_rstn and play_app_rstn.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLAGR_IEN15	FLAGR_IEN14	FLAGR_IEN13	FLAGR_IEN12	FLAGR_IEN11	FLAGR_IEN10	FLAGR_IEN9	FLAGR_IEN8	FLAGR_IEN7	FLAGR_IEN6	FLAGR_IEN5	FLAGR_IEN4	FLAGR_IEN3	FLAGR_IEN2	FLAGR_IEN1	FLAGR_IEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAGD_IEN15	FLAGD_IEN14	FLAGD_IEN13	FLAGD_IEN12	FLAGD_IEN11	FLAGD_IEN10	FLAGD_IEN9	FLAGD_IEN8	FLAGD_IEN7	FLAGD_IEN6	FLAGD_IEN5	FLAGD_IEN4	FLAGD_IEN3	FLAGD_IEN2	FLAGD_IEN1	FLAGD_IEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **FLAGR_IENy**: Flag interrupt enable for LEy registered output (y = 15 to 0)
0: No interrupt is generated on play_irq
1: An interrupt is generated on play_irq if PLAY_FLSTAT.FLAGRy = 1

Bits 15:0 **FLAGD_IENy**: Flag interrupt enable for LEy direct output (y = 15 to 0)
0: No interrupt is generated on play_irq
1: An interrupt is generated on play_irq if PLAY_FLSTAT.FLAGDy = 1

23.6.16 PLAY flag control register (PLAY_FLCTL)

Address offset: 0x230

Reset value: 0x0000 0000

This register selects the direction of signal transition to set the corresponding LE output flag. It is a functional register and is reset by play_rstn and play_app_rstn.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

Caution: play_clk must be active when writing to this register. Any subsequent write access to this register is ignored until the write completes. Software must poll the PLAY_FLCTL write busy flag in the *PLAY miscellaneous status register (PLAY_MSR)* to make sure that no writes are pending. If play_clk is slow compared to the bus clock, pclk, a write may take a long time to complete. In this case, the PLAY_FLCTL register write complete interrupt - see *PLAY interrupt status register (PLAY_ISR)* - can be used to signal to the software when the register can be accessed again.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLAGR_EDGE15	FLAGR_EDGE14	FLAGR_EDGE13	FLAGR_EDGE12	FLAGR_EDGE11	FLAGR_EDGE10	FLAGR_EDGE9	FLAGR_EDGE8	FLAGR_EDGE7	FLAGR_EDGE6	FLAGR_EDGE5	FLAGR_EDGE4	FLAGR_EDGE3	FLAGR_EDGE2	FLAGR_EDGE1	FLAGR_EDGE0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLAGD_EDGE15	FLAGD_EDGE14	FLAGD_EDGE13	FLAGD_EDGE12	FLAGD_EDGE11	FLAGD_EDGE10	FLAGD_EDGE9	FLAGD_EDGE8	FLAGD_EDGE7	FLAGD_EDGE6	FLAGD_EDGE5	FLAGD_EDGE4	FLAGD_EDGE3	FLAGD_EDGE2	FLAGD_EDGE1	FLAGD_EDGE0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **FLAGR_EDGEy**: Edge direction for LEy registered output (y = 15 to 0)
0: FLAGRy is set by a falling edge on leoutry
1: FLAGRy is set by a rising edge on leoutry

Bits 15:0 **FLAGD_EDGEy**: Edge direction for LEy direct output (y = 15 to 0)
0: FLAGDy is set by a falling edge on leoutdy
1: FLAGDy is set by a rising edge on leoutdy

23.6.17 PLAY miscellaneous status register (PLAY_MSR)

Address offset: 0x234

Reset value: 0x0000 0000

This register allows software to read the state of miscellaneous status flags (not including LE output flags). It is a functional register and is reset by play_rstn and play_app_rstn.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLCTL WBFS	SWIN WBFS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FLCTLWBFS**: PLAY_FLCTL register write busy flag status
0: PLAY_FLCTL register can be written
1: Write is pending, any new write access to PLAY_FLCTL register is ignored

Bit 0 **SWINWBFS**: PLAY_SWIN register write busy flag status
0: PLAY_SWIN register can be written
1: Write is pending, any new write access to PLAY_SWIN, PLAY_SWINCLR or PLAY_SWINSET registers is ignored

23.6.18 PLAY interrupt status register (PLAY_ISR)

Address offset: 0x238

Reset value: 0x0000 0000

This register allows software to read the state of the interrupt request sources. The PLAY_ISR register must be read first of all when the play_irq is asserted, to identify the source(s) of the interrupt request. It is a functional register and is reset by play_rstn and play_app_rstn.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLAGS	FLCTL WC	SWIN WC
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FLAGS**: Flag interrupts status
0: No flag interrupts active
1: One or more bits in PLAY_FLSTAT register is active and has generated an interrupt request

Bit 1 **FLCTLWC**: PLAY_FLCTL register write complete interrupt status
0: Interrupt request is not pending
1: Interrupt request is pending

Bit 0 **SWINWC**: PLAY_SWIN register write complete interrupt status
0: Interrupt request is not pending
1: Interrupt request is pending

23.6.19 PLAY interrupt clear register (PLAY_ICR)

Address offset: 0x23C

Reset value: 0xXXXX XXXX

This register allows software to clear active interrupt requests, except for those generated by flags - see *Section 23.6.14: PLAY flag clear register (PLAY_FLCLR)*. It is a functional register.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLCTL WC_CLR	SWIN WC_CLR
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FLCTLWC_CLR**: PLAY_FLCTL register write complete interrupt clear
0: No action
1: Clear interrupt request

Bit 0 **SWINWC_CLR**: PLAY_SWIN register write complete interrupt clear
0: No action
1: Clear interrupt request

23.6.20 PLAY interrupt enable register (PLAY_IER)

Address offset: 0x240

Reset value: 0x0000 0000

This register allows software to enable or disable interrupt requests, except for those generated by flags - see *Section 23.6.15: PLAY flag interrupt enable register (PLAY_FLIER)*. It is a functional register and is reset by play_rstn and play_app_rstn.

If PLAY_SECCFGR.SEC[1] = 1, the register requires secure access (nonsecure accesses are ignored).

If PLAY_PRIVCFGR.PRIV[1] = 1, the register requires privileged access (unprivileged accesses are ignored).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLCTLWC_IEN	SWINWC_IEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FLCTLWC_IEN**: PLAY_FLCTL register write complete interrupt enable
0: Interrupt disabled
1: Interrupt enabled. An interrupt is generated when a write to the PLAY_FCTL register completes

Bit 0 **SWINWC_IEN**: PLAY_SWIN register write complete interrupt enable
0: Interrupt disabled
1: Interrupt enabled. An interrupt is generated when a write to the PLAY_SWIN register completes

23.7 PLAY register map

Table 209. PLAY register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PLAY_CFGCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK
	Reset value																																1
0x004	Reserved	Res.																															

Table 209. PLAY register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	PLAY_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC[1:0]	
	Reset value																															0	0
0x008	Reserved	Res.																															
0x008	PLAY_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV[1:0]	
	Reset value																															0	0
0x00C to 0x01C	Reserved	Res.																															
0x020 + 0x4 * x, (x=0 to 15) Last address: 0x05C	PLAY_FILTxCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREMUXSEL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	EDGEDET [1:0]		WIDTH [7:0]							
	Reset value														0	0	0							0	0	0	0	0	0	0	0	0	0
0x060 + 0x4 * x, (x=0 to 15) Last address: 0x09C	PLAY_LExCFG1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0 + 0x4 * x, (x=0 to 15) Last address: 0x0DC	PLAY_LExCFG2	Res.	Res.	CK_SEL[4:0]						IN3_SEL[5:0]						IN2_SEL[5:0]						IN1_SEL[5:0]					IN0_SEL[5:0]						
	Reset value			0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0E0 - 0x0FC	Reserved	Res.																															
0x100 + 0x4 * x, (x=0 to 15) Last address: 0x13C	PLAY_OUTxCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEL[4:0]				
	Reset value																												0	0	0	0	0
0x140 - 0x1FC	Reserved	Res.																															
0x200	PLAY_SWIN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIN15	SWIN14	SWIN13	SWIN12	SWIN11	SWIN10	SWIN9	SWIN8	SWIN7	SWIN6	SWIN5	SWIN4	SWIN3	SWIN2	SWIN1	SWIN0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x204	PLAY_SWINSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIN15	SWIN14	SWIN13	SWIN12	SWIN11	SWIN10	SWIN9	SWIN8	SWIN7	SWIN6	SWIN5	SWIN4	SWIN3	SWIN2	SWIN1	SWIN0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208	PLAY_SWINCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIN15	SWIN14	SWIN13	SWIN12	SWIN11	SWIN10	SWIN9	SWIN8	SWIN7	SWIN6	SWIN5	SWIN4	SWIN3	SWIN2	SWIN1	SWIN0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20C	Reserved	Res.																															
0x210	PLAY_OSR	LEOUTR15	LEOUTR14	LEOUTR13	LEOUTR12	LEOUTR11	LEOUTR10	LEOUTR9	LEOUTR8	LEOUTR7	LEOUTR6	LEOUTR5	LEOUTR4	LEOUTR3	LEOUTR2	LEOUTR1	LEOUTR0	LEOUTD15	LEOUTD14	LEOUTD13	LEOUTD12	LEOUTD11	LEOUTD10	LEOUTD9	LEOUTD8	LEOUTD7	LEOUTD6	LEOUTD5	LEOUTD4	LEOUTD3	LEOUTD2	LEOUTD1	LEOUTD0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x214 - 0x21C	Reserved	Res.																															

Table 209. PLAY register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x220	PLAY_FLSTAT	FLAGR15	FLAGR14	FLAGR13	FLAGR12	FLAGR11	FLAGR10	FLAGR9	FLAGR8	FLAGR7	FLAGR6	FLAGR5	FLAGR4	FLAGR3	FLAGR2	FLAGR1	FLAGR0	FLAGD15	FLAGD14	FLAGD13	FLAGD12	FLAGD11	FLAGD10	FLAGD9	FLAGD8	FLAGD7	FLAGD6	FLAGD5	FLAGD4	FLAGD3	FLAGD2	FLAGD1	FLAGD0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x224	PLAY_FLSET	FLAGR_SET15	FLAGR_SET14	FLAGR_SET13	FLAGR_SET12	FLAGR_SET11	FLAGR_SET10	FLAGR_SET9	FLAGR_SET8	FLAGR_SET7	FLAGR_SET6	FLAGR_SET5	FLAGR_SET4	FLAGR_SET3	FLAGR_SET2	FLAGR_SET1	FLAGR_SET0	FLAGD_SET15	FLAGD_SET14	FLAGD_SET13	FLAGD_SET12	FLAGD_SET11	FLAGD_SET10	FLAGD_SET9	FLAGD_SET8	FLAGD_SET7	FLAGD_SET6	FLAGD_SET5	FLAGD_SET4	FLAGD_SET3	FLAGD_SET2	FLAGD_SE1	FLAGD_SET0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x228	PLAY_FLCLR	FLAGR_CLR15	FLAGR_CLR14	FLAGR_CLR13	FLAGR_CLR12	FLAGR_CLR11	FLAGR_CLR10	FLAGR_CLR9	FLAGR_CLR8	FLAGR_CLR7	FLAGR_CLR6	FLAGR_CLR5	FLAGR_CLR4	FLAGR_CLR3	FLAGR_CLR2	FLAGR_CLR1	FLAGR_CLR0	FLAGD_CLR15	FLAGD_CLR14	FLAGD_CLR13	FLAGD_CLR12	FLAGD_CLR11	FLAGD_CLR10	FLAGD_CLR9	FLAGD_CLR8	FLAGD_CLR7	FLAGD_CLR6	FLAGD_CLR5	FLAGD_CLR4	FLAGD_CLR3	FLAGD_CLR2	FLAGD_CLR1	FLAGD_CLR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x22C	PLAY_FLIER	FLAGR_IEN15	FLAGR_IE14	FLAGR_IEN13	FLAGR_IEN12	FLAGR_IEN11	FLAGR_IEN10	FLAGR_IEN9	FLAGR_IEN8	FLAGR_IEN7	FLAGR_IEN6	FLAGR_IEN5	FLAGR_IEN4	FLAGR_IEN3	FLAGR_IEN2	FLAGR_IEN1	FLAGR_IEN0	FLAGD_IEN15	FLAGD_IEN14	FLAGD_IEN13	FLAGD_IEN12	FLAGD_IEN11	FLAGD_IEN10	FLAGD_IEN9	FLAGD_IEN8	FLAGD_IEN7	FLAGD_IEN6	FLAGD_IEN5	FLAGD_IEN4	FLAGD_IEN3	FLAGD_IEN2	FLAGD_IEN1	FLAGD_IEN0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x230	PLAY_FLCTL	FLAGR_EDGE15	FLAGR_EDGE14	FLAGR_EDGE13	FLAGR_EDGE12	FLAGR_EDGE11	FLAGR_EDGE10	FLAGR_EDGE9	FLAGR_EDGE8	FLAGR_EDGE7	FLAGR_EDGE6	FLAGR_EDGE5	FLAGR_EDGE4	FLAGR_EDGE3	FLAGR_EDGE2	FLAGR_EDGE1	FLAGR_EDGE0	FLAGD_EDGE15	FLAGD_EDGE14	FLAGD_EDGE13	FLAGD_EDGE12	FLAGD_EDGE11	FLAGD_EDGE10	FLAGD_EDGE9	FLAGD_EDGE8	FLAGD_EDGE7	FLAGD_EDGE6	FLAGD_EDGE5	FLAGD_EDGE4	FLAGD_EDGE3	FLAGD_EDGE2	FLAGD_EDGE1	FLAGD_EDGE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x234	PLAY_MSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLCTLWBFS	SWINWBFS
	Reset value																															0	0
0x238	PLAY_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLAGS	FLCTLWC	SWINWC
	Reset value																														0	0	0
0x23C	PLAY_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLCTLWC_CLR	SWINWC_CLR
	Reset value																															X	X
0x240	PLAY_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLCTLWC_IEN	SWINWC_IEN
	Reset value																															0	0
0x244 - 0x3EC	Reserved	Res.																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

24 Extended-SPI interface (XSPI)

This section applies only for STM32G55x/56x devices.

24.1 XSPI introduction

The XSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- indirect mode: all the operations are performed using the XSPI registers to preset commands, addresses, data, and transfer parameters.
- automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The XSPI supports the following protocol with associated frame format:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

24.2 XSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- External (P)SRAM memory support
- Support for single, dual, quad, octal communication
- Dual-memory configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- Asynchronous bus clock versus kernel clock support
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- XSPI interface with transaction acceptance limited to one: the interface accepts the next transfer on AHB bus only once the previous is completed on memory side.

- Dual chip select support (NCS1 and NCS2)
- Extended external memory support: if two same size external memories (extmem1 and extmem2) are connected to the same I/O port, contiguously in the memory map and driven by a single XSPI. This XSPI automatically switches CS to extmem1 or extmem2, according to the address on interconnect side.
- Possibility to disable the automatic prefetch

24.3 XSPI implementation

Table 210. XSPI implementation

XSPI feature	XSPI1
Octal mode configuration (8-bit)	X
16-bit mode configuration	-
HyperBus standard compliant	X
Xccela standard compliant	X
XSPI (JESD251C) standard compliant	X
AMBA® AHB compliant data interface	X
Dual AHB interface	X
Asynchronous AHB clock versus kernel clock	X
Functional modes: indirect, automatic status-polling, and memory-mapped(1)	X
Read and write support in memory-mapped mode	X
Dual-quad configuration	X
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS,DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8-, 16-, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Bus error bit	-
Address offset	-
Extended CSHT timeout	X
Memory-mapped write	X
Refresh counter	X
GPDMA interface	X
Dual chip select	X
Extended external memory	X

Table 210. XSPI implementation (continued)

XSPI feature	XSPI1
Prefetch disable	X
Prefetch hardware software	-

1. Automatic status polling mode is not supported.

24.4 XSPI functional description

24.4.1 XSPI block diagram

Figure 169. XSPI block diagram for octal configuration

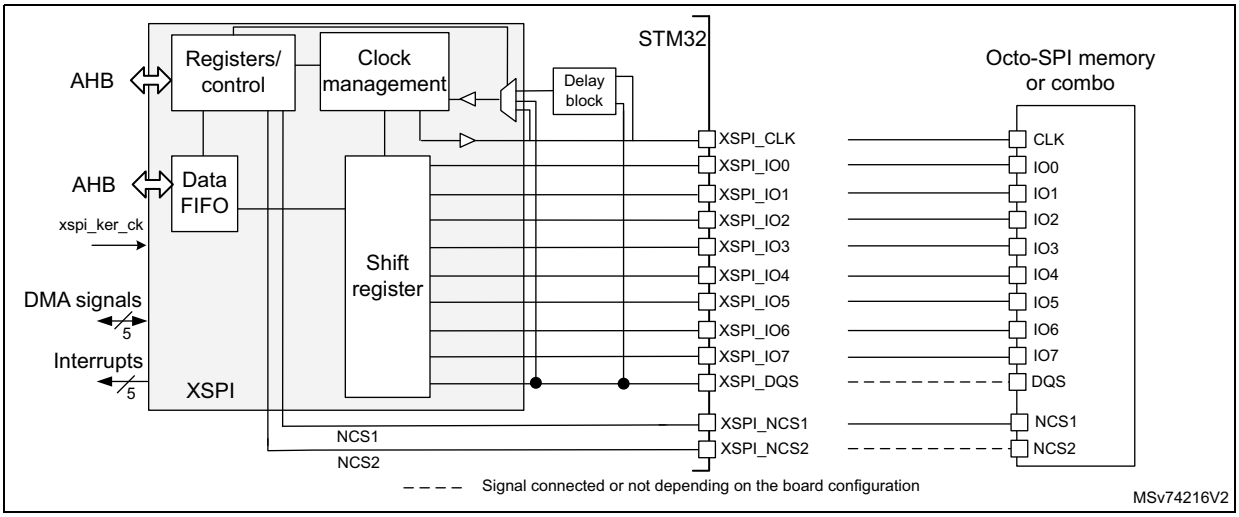


Figure 170. XSPI block diagram for quad configuration

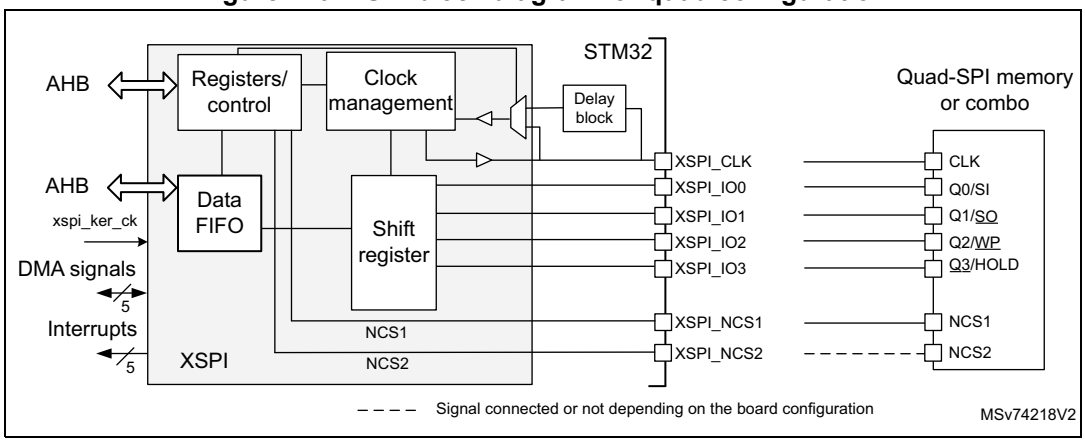
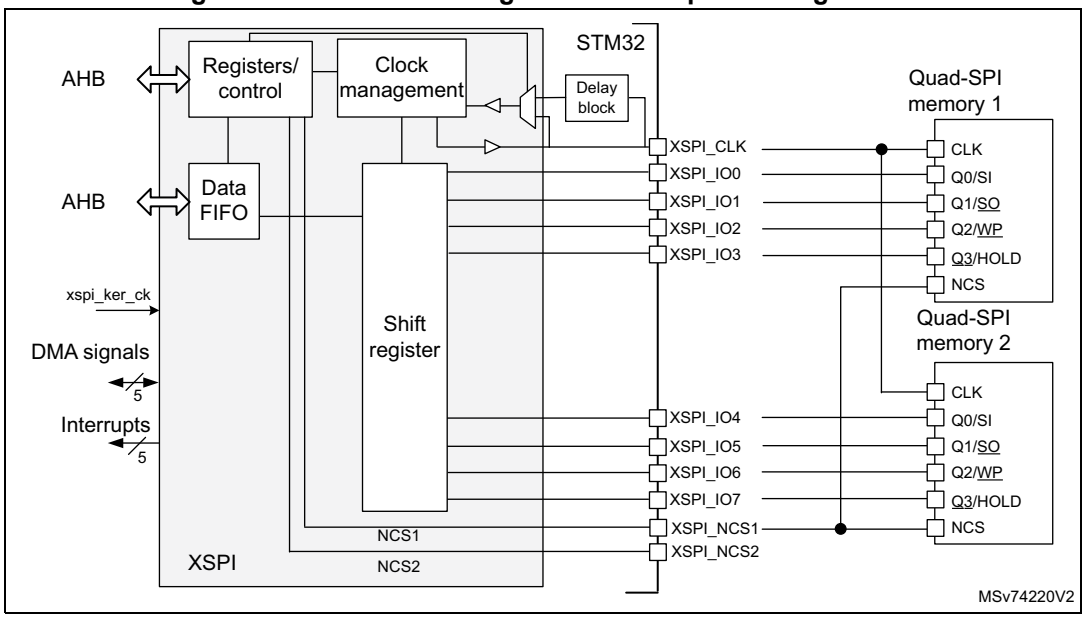


Figure 171. XSPI block diagram for dual-quad configuration



24.4.2 XSPI pins and internal signals

Table 211. XSPI input/output pins

Pin name	Type	Description
XSPI_CLK	Output	XSPI clock
XSPI_IOn (n = 0 to 7)	Input/output	XSPI data pins
XSPI_NCS1,2	Output	Chip select for the memory
XSPI_DQS	Input/output	Data strobe/write mask signal from/to the memory

Caution: Use the same configuration (output speed, HSLV[(a)]) for all XSPI input/output pins to avoid any data corruption.

Table 212. XSPI internal signals

Signal name	Type	Description
xspi_hclk	Input	XSPI AHB clock
xspi_ker_ck	Input	XSPI kernel clock
xspi_dma	N/A	DMA request signal
xspi_it	Output	Global interrupt line (see *Table 216* for the multiple sources of interrupt)

a. When applicable.

24.4.3 XSPI interface to memory modes

The XSPI supports the following protocol:

- regular-command protocol
- HyperBus protocol

The XSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select (in the following sections, unless otherwise specified, NCS applies for NCS1 or NCS2. Only one of these two signals is active at a given moment in time)
- CLK: communication clock
- DQS: data strobe used only in regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus can be used as possible remap for quad-SPI mode

24.4.4 XSPI regular-command protocol

When in regular-command protocol, the XSPI communicates with the external device using commands. Each command can include the following phases:

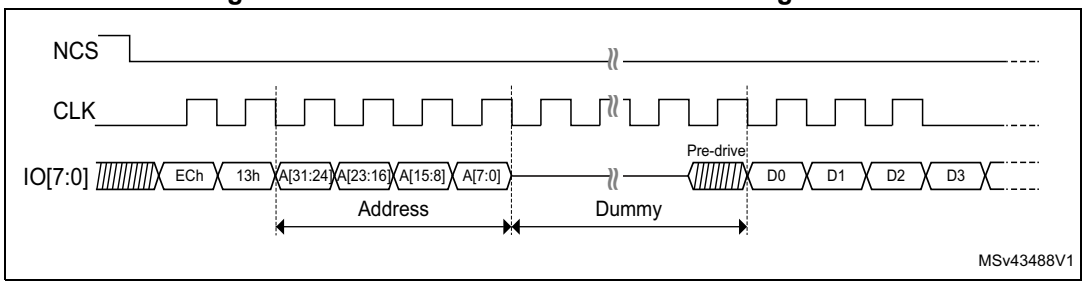
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operations are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 172. SDR read command in octal configuration



The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured by ISIZE[1:0] in XSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of XSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using IMODE[2:0] of XSPI_CCR.

The instruction can be sent in DTR mode on each rising and falling edge of the clock, by setting IDTR in XSPI_CCR.

When IMODE[2:0] = 000 in XSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In memory-mapped mode, the instruction used for the write operation is specified in XSPI_WIR, and the instruction format is specified in XSPI_WCCR. The instruction used for the read operation and the instruction format are specified in XSPI_IR and XSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured by ADSIZE[1:0] in XSPI_CCR.

In indirect and automatic status-polling modes, the address bytes to be sent are specified by ADDRESS[31:0] in XSPI_AR. In memory-mapped mode, the address is given directly via the AHB (from any master in the system).

The address phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ADMODE[2:0] in XSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in XSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in XSPI_WCCR. The address format for the read operation is specified in XSPI_CCR.

Alternate-byte phase

In the alternate-byte phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured by ABSIZE[1:0] in XSPI_CCR. The bytes to be sent are specified in XSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ABMODE[2:0] in XSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR in XSPI_CCR.

When ABMODE[2:0] = 000, the alternate-byte phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in XSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in XSPI_WABR, and the alternate byte format is specified in XSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in XSPI_ABR and XSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified by DCYC[4:0] in XSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-, the quad-, or the octal-SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in XSPI_WTCR. The dummy cycles for the read operation are specified in XSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in XSPI_DLR. In this mode, the data to be sent to the external device must be written to XSPI_DR, while in indirect-read mode the data received from the external device is obtained by reading XSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in XSPI_DLR, and the data received from the external device can be obtained by reading XSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AHB to the Cortex core or to a DMA.

The data phase can send/receive:

- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using DMODE[2:0] in XSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR in XSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

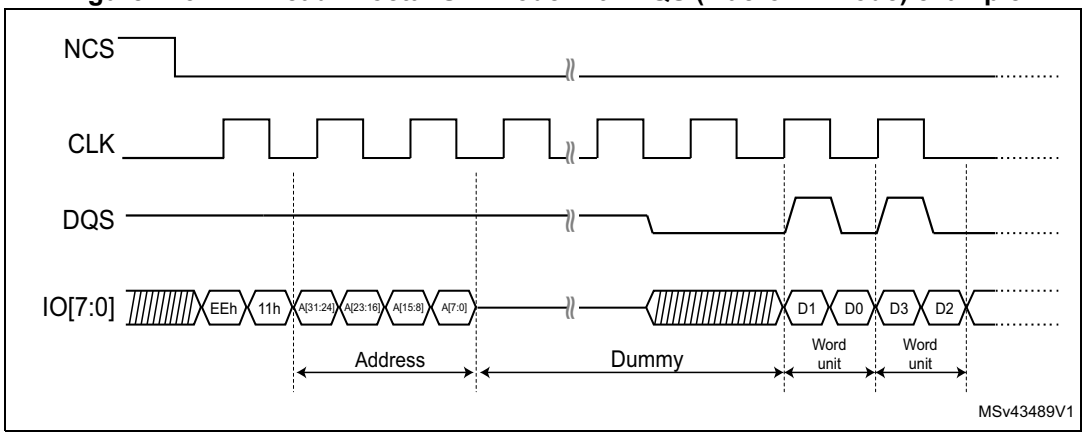
In memory-mapped mode, the data format for the write operation is specified in XSPI_WCCR. The data format for the read operation is specified in XSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE in XSPI_CCR.

Figure 173. DTR read in octal-SPI mode with DQS (Macronix mode) example



24.4.5 XSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, I0/SO and I1/SI are respectively equivalent to MOSI and MISO, having the XSPI generating the clock.

The different phases can each be configured separately to use this single-SPI mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 010, IO0 and IO1 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 011, IO0 to IO3 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase, when DMODE[2:0] = 100, IO[0:7] are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

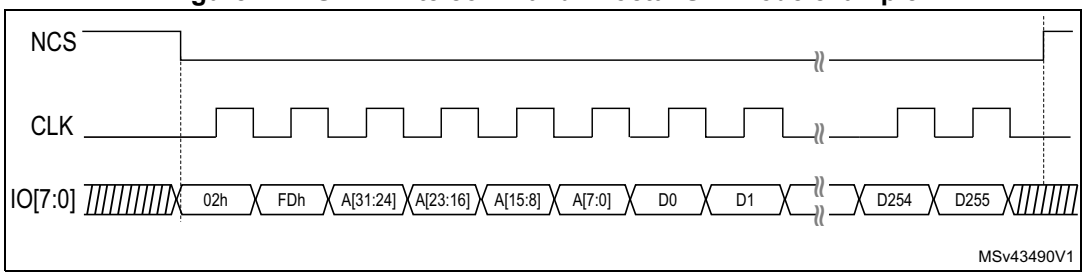
Single-data rate (SDR) mode

By default, all the phases operate in single-data rate (SDR) mode.

In SDR mode, when the XSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the XSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in XSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 174. SDR write command in octal-SPI mode example



Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in XSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the write operations is specified in XSPI_WCCR. The DTR mode for each phase of the read operations is specified in XSPI_CCR.

In DTR mode, when the XSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

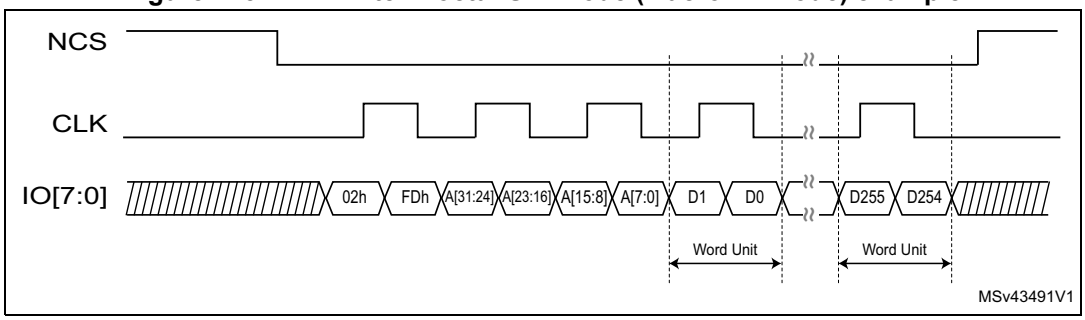
In DTR mode, when the XSPI drives the IO0 to IO7 signals in the data phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the XSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in XSPI_CCR, the software must clear SSHIFT in XSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of XSPI_TCR, to shift the outputs by a quarter of cycle and avoid holding issues on the memory side.

Note: DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

Figure 175. DTR write in octal-SPI mode (Macronix mode) example



Dual-quad configuration

When DMM = 1 in XSPI_CR, the XSPI is in dual-memory configuration: if DMODE = 011, two external quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified by DEVSIZE[4:0] in XSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the XSPI gives for address X is the byte at the address X/2 of device A, and the byte that the XSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the XSPI must be configured with a data length of 2 bytes (16 bits), and the XSPI receives one byte from each device.

If each device gives a status of 16 bits, then the XSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The fourth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in XSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

24.4.6 HyperBus protocol

The XSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- DQS for read and write data strobe and latency insertion
- NCS
- CLK

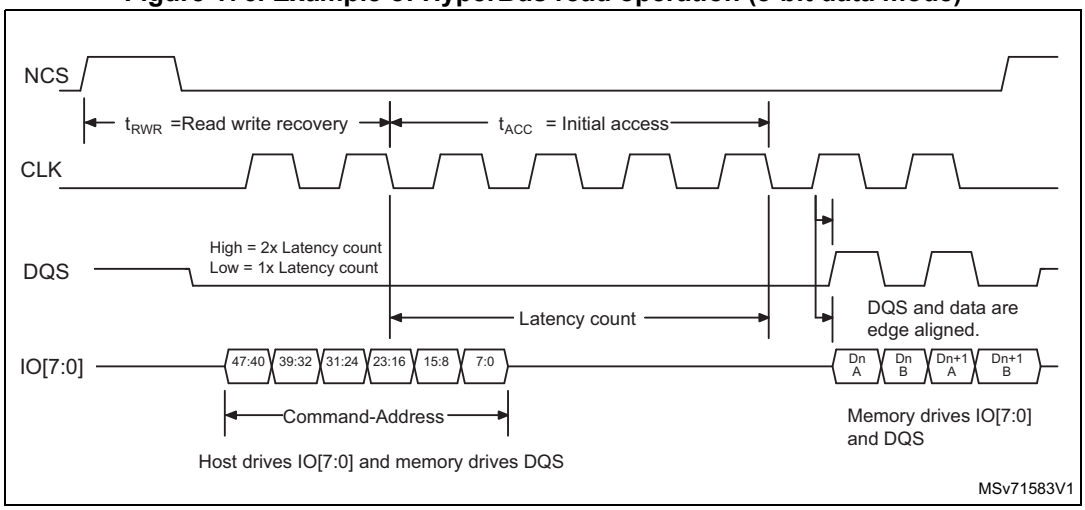
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- Command/address phase
- Data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 176. Example of HyperBus read operation (8-bit data mode)



The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the XSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 213. Command/address phase description

CA bit	Bit name	Description
47	R/W#	Identifies the transaction as a read or a write.
46	Address space	Indicates if the transaction accesses the memory or the register space.

Table 213. Command/address phase description

CA bit	Bit name	Description
45	Burst type	Indicates if the burst is linear or wrapped.
44-16	Row and upper column address	Selects the row and the upper column addresses.
15-3	Reserved	-
2-0	Lower column address	Selects the starting 16-bit word within the half page.

The address space is configured through the memory type MTYP[2:0] in XSPI_DCR1.

The total size of the device must be configured through DEVSIZE[4:0] of XSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:

- t_{RWR}: minimal read/write recovery time for the device (defined by TRWR[7:0] in XSPI_HLCR)
- t_{ACC}: access time for the device (defined by TACC[7:0] in XSPI_HLCR) according to the memory latency

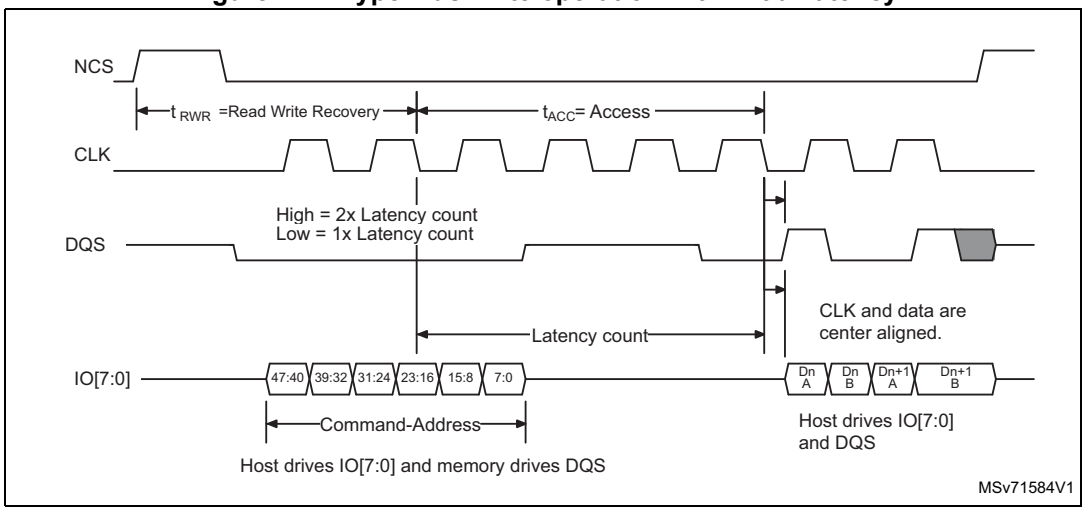
During the read operation, the DQS is used by the device, in two ways (see *Figure 176*):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation, the DQS is used:

- by the device, during the command/address phase, to request an additional latency.
- by the XSPI, during the data phase, for write data masking.

Figure 177. HyperBus write operation with initial latency



Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), DQS must be tied to one during one of the DQS signals, during the command/address phase.

An additional t_{ACC} duration is added by the XSPI to meet the device request.

Figure 178. HyperBus read operation with additional latency

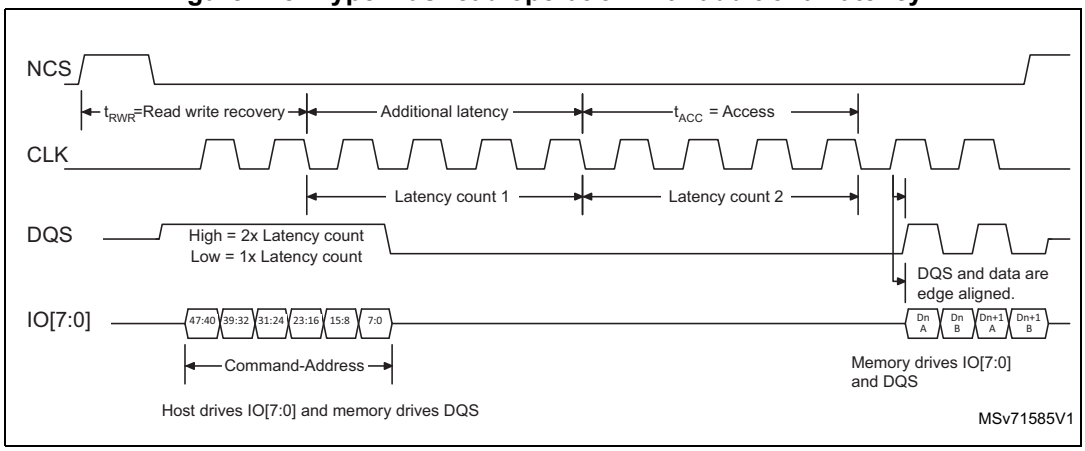
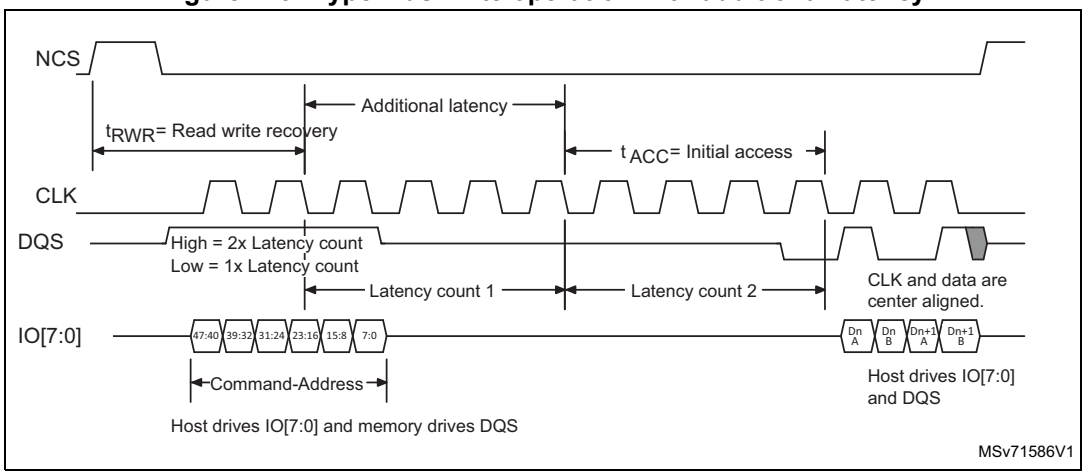


Figure 179. HyperBus write operation with additional latency



Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

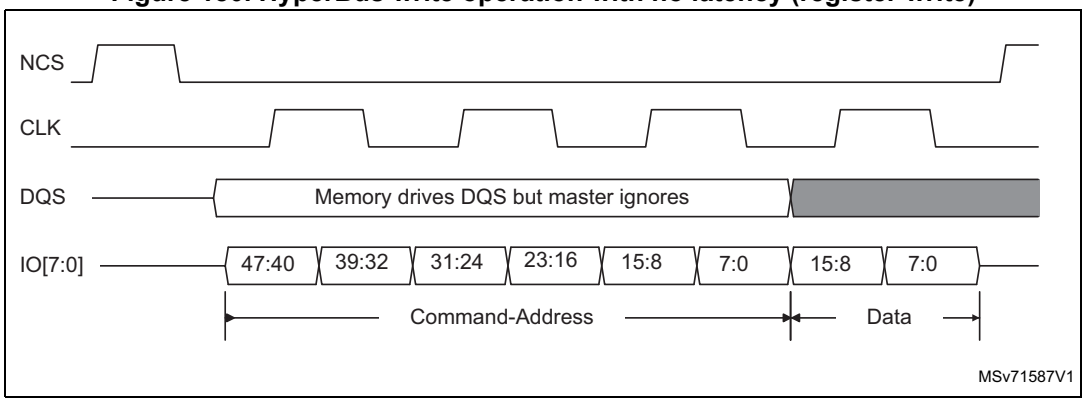
The latency can be forced to 2 x t_{ACC} by setting LM in XSPI_HLCR.

In this XSPI latency mode, the state of the DQS signal is not taken into account by the XSPI, and an additional latency is always added, leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in XSPI_HLCR.

Figure 180. HyperBus write operation with no latency (register write)

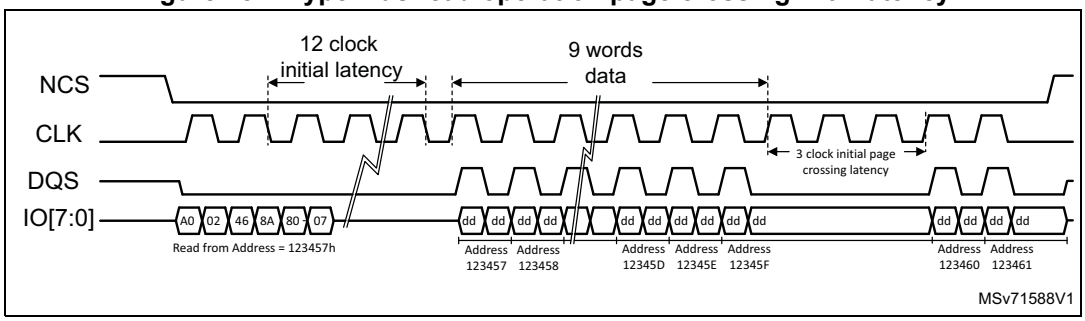


Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t_{ACC} time.

Figure 181. HyperBus read operation page crossing with latency



24.4.7 Specific features

The XSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh

Wrap support

The XSPI supports a hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in XSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 4 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated XSPI_WPxxx registers must be programmed according to the wrapped-read access characteristics. These registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: Hybrid wrap requires that the nonwrapped registers (XSPI_CCR, XSPI_TCR, XSPI_IR) are set according to the memory configuration to satisfy its correct data prefetch (initiated after the wrap command).

The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.

NCS boundary and refresh

Two processes can be activated to regulate the XSPI transactions:

- NCS boundary
- Refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured by CSBOUND[4:0] in XSPI_DCR3: it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND(4:0] = 0x4, the boundary is set to 2^4 = 16 bytes. The NCS is then released each time the LSB address is equal to 0xF, and each time a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.

The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed by REFRESH[31:0] in XSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

24.4.8 XSPI operating mode introduction

The XSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):

- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

24.4.9 XSPI indirect mode

In indirect mode, the commands are started by writing to the XSPI registers, and data are transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in XSPI_CR, the XSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to XSPI_DR.

When FMODE[1:0] = 01, the XSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading XSPI_DR.

In indirect mode, when the XSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The XSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in XSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the XSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in XSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the XSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, the TCF bit is set in XSPI_SR, and an interrupt is generated if TCIE = 1 in XSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in XSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the XSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using the regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data need to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in XSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data need to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in XSPI_AR
- if data need to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in XSPI_DR

A write to XSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in XSPI_SR.

Triggering the start of a transfer in HyperBus protocol

Depending on the XSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication

in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (XSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01).
- when a write is performed to DATA[31:0] (XSPI_DR) in indirect-write mode (when FMODE = 00).
- when a (dummy) write is performed to INSTRUCTION[31:0] (XSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01).

As soon as a transfer is started, the BUSY bit (XSPI_SR[5]) is automatically set.

FIFO and data management

Data in indirect mode pass through a 32-byte FIFO that is internal to the XSPI. FLEVEL in XSPI_SR indicates how many bytes are currently being held in the FIFO.

AHB burst transactions are supported. Data of the burst are successively written in XSPI_DR, and immediately transferred in the internal FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in XSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to XSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

24.4.10 XSPI automatic status-polling mode

In automatic status-polling mode, the XSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value. The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in XSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in XSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in XSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in XSPI_CR, the AND-match mode is activated: SMF is set in XSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in XSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in XSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the XSPI is disabled (EN = 0).

XSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in XSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. As a consequence, the memory must then be configured in fixed latency.

24.4.11 XSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256-Mbyte range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the XSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the XSPI never stops its prefetch operation. It either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time. This prefetch feature can be disabled by setting the NOPREF bit in XSPI_DCR1. Disabling the prefetch can be beneficial for performance when the transfer requests are short and jumping frequently between non-consecutive addresses.

Since external devices tend to consume more power when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in XSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in XSPI_LPTR, when x cycles have elapsed without access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes. The indirect-write mode fulfills this operation.

However, if the application requires the use of the MCE for encryption (check MCE product availability), the memory-mapped write mode may be used to program encrypted data to external flash memory under the following conditions:

- Prefetch must be enabled.
- In block cipher mode, the CPU must write a complete 128-bit data block to prevent the MCE from initiating read-modify-write operations when only a few bytes need to be programmed. This precaution avoids incorrect programming operations. There are no specific constraints to respect if the MCE is used in stream cipher mode.
- Apply the abort sequence to exit memory-mapped mode when the data linked to the page has been written in the external memory buffers. The abort sequence triggers the start of the page programming.
- Switch to the automatic status-polling mode to monitor the completion of the page programing phase.
- Relaunch the write enable command in indirect mode, then switch back to the memory-mapped mode configuration to continue to program additional pages if any.

It is recommended to add a synchronization barrier between the end of the controller registers configuration and the first memory-mapped access to the external memory when the controller is configured in memory-mapped mode.

24.4.12 XSPI configuration introduction

The XSPI configuration is done in three steps:

1. XSPI system configuration
2. XSPI device configuration
3. XSPI mode configuration

24.4.13 XSPI system configuration

The XSPI is configured using XSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
- the DMA use with DMAEN
- the timeout counter use with TCEN
- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in XSPI_LPTR.

The DMA channel must not be enabled during the XSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and XSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

24.4.14 XSPI device configuration

The parameters related to the external device targeted are configured through XSPI_DCR1 and XSPI_DCR2.The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the clock mode with FRCK and CKMODE
- the associated delay block DLYB connected externally to the peripheral
- the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{[\text{DEVSIZE}+1]}$$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the XSPI executes two commands, one immediately after the other, it raises the chip-select signal (NCS) high between the two commands for at least one CLK cycle by default.

If the external device requires more time between commands, CSHT[5:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (t_{ACC} and t_{RWR}) and the latency mode must be configured in XSPI_HLCR.

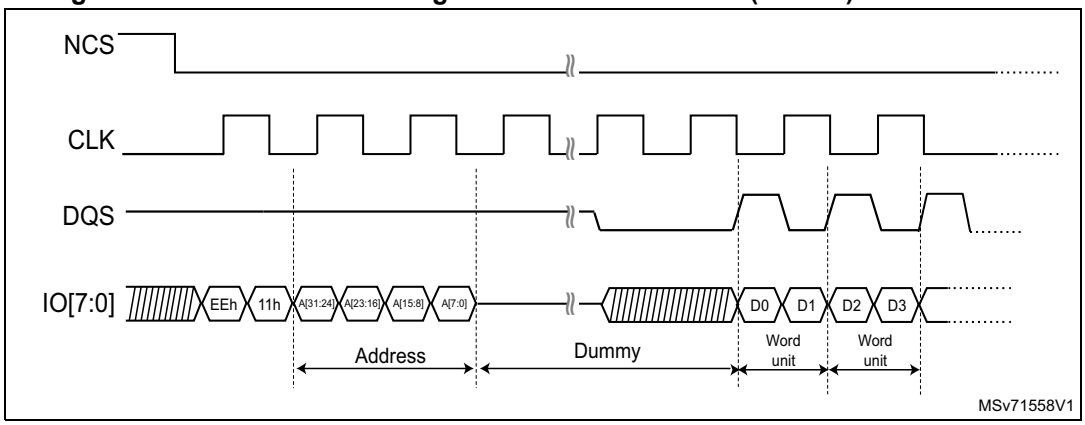
Memory types

External memory providers may present some architecture and slight data formatting differences between them. The bitfield MTYP[2:0] into the XSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. This is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in a such way:

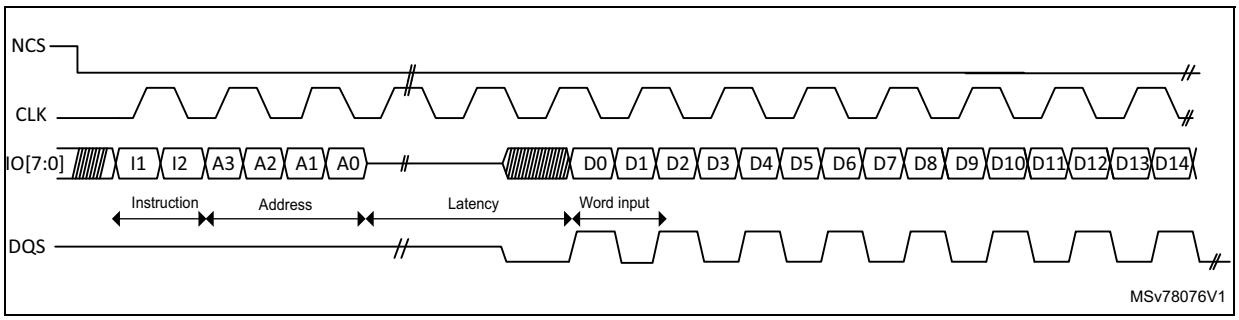
- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

Figure 182. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the memory. Refer to *Figure 173: DTR read in octal-SPI mode with DQS (Macronix mode) example*.
- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 010. DQS is keeping the same polarity as the clock when reading data from the memory. It is defined as the standard mode since it is the configuration to select when the targeted memory does not fit with any others modes proposed by the MTYP configuration values.

Figure 183. D0/D1 data ordering in octal-SPI DTR mode with DQS and CLK polarity aligned during Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to *Table 214: OctaRAM command address bit assignment (based on 64 Mb OctaRAM)*. This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock one when reading data from the memory.

Figure 184. OctaRAM read operation with reverse data ordering D1/D0

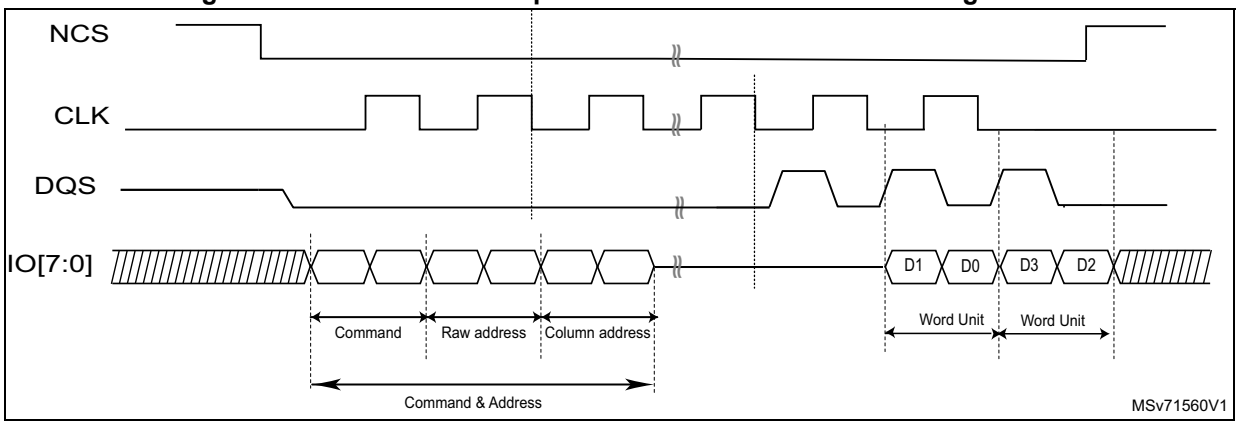


Table 214. OctaRAM command address bit assignment (based on 64 Mb(1) OctaRAM)

Clock	1st clock	2nd clock		3rd clock	
Function	Command	Row address		Column address	
SIO[7]	Command	Reserved	RA7	CA9	Reserved
SIO[6]		Reserved	RA6	CA8	Reserved
SIO[5]		Reserved	RA5	CA7	Reserved
SIO[4]		RA12	RA4	CA6	Reserved
SIO[3]		RA11	RA3	CA5	CA3
SIO[2]		RA10	RA2	CA4	CA2
SIO[1]		RA9	RA1	Reserved	CA1
SIO[0]		RA8	RA0	Reserved	CA0(2)

1. Example of 64 Mb OctaRAM address assignment:
   Row Address [RA12:RA0]: 8K. Column Address [CA9:CA0]: 1K. 64 Mb density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- HyperBus memories need to be selected when targeted by the application. The configuration to set depends on the access type:
  - HyperBus memory mode: The protocol follows the HyperBus specification. MTYP[2:0] = 100 is the configuration to use to access the memory space.
  - HyperBus register mode (addressing register space): the memory-mapped accesses in this mode must be noncacheable, or the indirect read/write modes must be used. The configuration to be set for this particular register space access is MTYP[2:0] = 101.

24.4.15 XSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected, and data can be read from the external device.

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the frame timing in XSPI_TCR.
3. Specify the frame format in XSPI_CCR.
4. Specify the instruction in XSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
6. Specify the targeted address in XSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

If neither the address register (XSPI_AR) nor the data register (XSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as XSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the communication start is triggered by a write in the FIFO through XSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and data are retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in XSPI_DLR, it is ignored, and only 4 bytes are read. The periodicity is specified in XSPI_PIR.

Once the status data has been retrieved, the following can be processed:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in XSPI_PSMKR, and can be ORed or ANDed with the value stored in XSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The XSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in XSPI_DR.

When the XSPI is used in automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in XSPI_PSMKR.
2. Specify the comparison value in XSPI_PSMAR.

3. Specify the read period in XSPI_PIR.
4. Specify a number of data bytes to read in XSPI_DLR.
5. Specify the frame timing in XSPI_TCR.
6. Specify the frame format in XSPI_CCR.
7. Specify the instruction in XSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
9. Specify the optional targeted address in XSPI_AR.

If the address register (XSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as XSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

The memory-mapped mode is entered by setting FMODE[1:0] = 11 in XSPI_CR.

The programmed instruction and frame are sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

When the XSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in XSPI_TCR for read operation.
2. Specify the frame format in XSPI_CCR for read operation.
3. Specify the instruction in XSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR for read operation.
5. Specify the frame timing in XSPI_WTCR for write operation.
6. Specify the frame format in XSPI_WCCR for write operation.
7. Specify the instruction in XSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_WABR for write operation.

All configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 has no effect and is not signaled with an error response. On the first access, the XSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

XSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the XSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in XSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

XSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it is shifted precisely by one quarter of the SPI clock cycle, for all frequencies above freq_min, to compensate the data propagation delay in the “high-speed interface” when the product embeds one.

This shift is performed by an external delay block (DLYB) located outside the XSPI. This external delay block has to be calibrated (by enabling FRCK) before starting the first transfer, and recalibrated regularly. The control of this feature depends on the device implementation (see the delay block (DLYB) section in this document for more details about DLYB and calibration procedure, and the product datasheet for freq_min value).

In configurations where delay does not need to be compensated, the external delay block can be bypassed by setting DLYBYP in XSPI_DCR1.

24.4.16 XSPI HyperBus protocol configuration

Indirect mode configuration (HyperBus)

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the targeted address in XSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through XSPI_DR.

Memory-mapped mode configuration (HyperBus)

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the XSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit, or to abort by setting the ABORT bit.

24.4.17 XSPI error management

The following errors set the TEF flag in XSPI_SR and generates an interrupt if enabled (TEIE = 1 in XSPI_CR):

- in indirect or automatic status-polling mode, when a wrong address has been programmed in XSPI_AR (according to the device size defined by DEVSIZE[4:0]).
- in indirect mode, if the address plus the data length exceed the device size: TEF is set as soon as the access is triggered.

In memory-mapped mode, the XSPI generates an AHB slave error in the following situations, setting the BERRF flag in XSPI_SR and generating an interrupt if enabled (BERRIE = 1 in XSPI_CR) (if the Bus error bit feature is available in the product implementation table, otherwise the Slave bus error response is generated at system level as transaction response with no dedicated bit into the controller status register):

- The memory-mapped mode is disabled and an AHB read or write request occurs.
- A read or write address exceeds the size of the external memory.
- An abort is received while a read or write burst is ongoing when the abort condition is present at the moment of transfer phases preceding the data phase (In incremental or wrap burst type modes). To avoid the error, refer to *Section 24.4.18: XSPI BUSY and ABORT*.
- The XSPI is disabled while a read or write burst is requested.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in XSPI_WCCR in octal DTR mode, in dual-memory configuration, in HyperBus mode or 16-bit mode.
- A read or write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means that the AHB bus transfer size (HSIZE) is different from 4 bytes (only for memory-mapped mode).
- Bit DMM is set in XSPI_CR (dual-memory configuration) and octal mode is set.

24.4.18 XSPI BUSY and ABORT

Once the XSPI starts an operation with the external device, BUSY is automatically set in XSPI_SR.

In indirect mode, BUSY is reset once the XSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in XSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in XSPI_DR, it is recommended to disable the DMA channel before aborting the SPI.

Note: Some devices may misbehave if a write operation to a status register is aborted.

24.4.19 XSPI reconfiguration or deactivation

Before any XSPI reconfiguration, the software must ensure that all the transactions are completed:

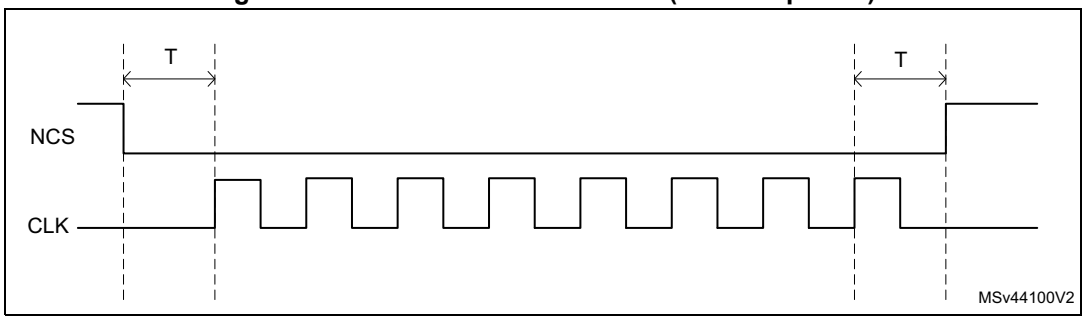
- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier than an abort.

24.4.20 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 185. NCS when CKMODE = 0 (T = CLK period)



Note: Clock mode 3 is not supported.

In dual-memory configuration, the NCS2 behaves exactly the same as the NCS1.
The quad-SPI/octal-SPI memory 2 can then use the NCS1, and the NCS2 pin can be used for other functions.

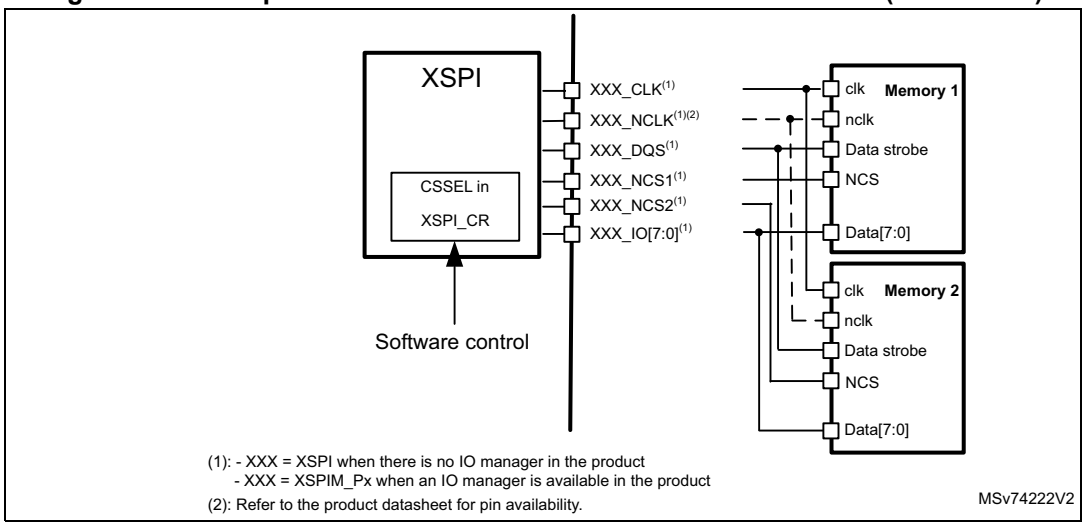
24.4.21 Software control of two external memories

One XSPI instance is able to control the generation of the two chip-select signals (NCS1 and NCS2). This allows two external memories to be connected to the same I/O port driven by the XSPI.

The XSPI generates one NCS signal and the user defines how the NCS is directed to NCS1 or NCS2, by setting the CSSEL bit in XSPI_CR. Intended operation is semistatic commutation, such as starting an application reading an external flash memory and then, for the rest of the application, use an external SRAM connected to the same I/O bus.

From a memory-map point of view, these two memories are located in the same address range.

Figure 186. Example of software control of two external memories (octal mode)



24.4.22 Hardware-controlled extended memory support

One XSPI instance is able to control autonomously the generation of the two chip-select signals (NCS1 and NCS2). This feature is activated by setting EXTENDMEM in XSPI_DCR1 and allows two external memories to be connected to the same I/O port driven by the XSPI.

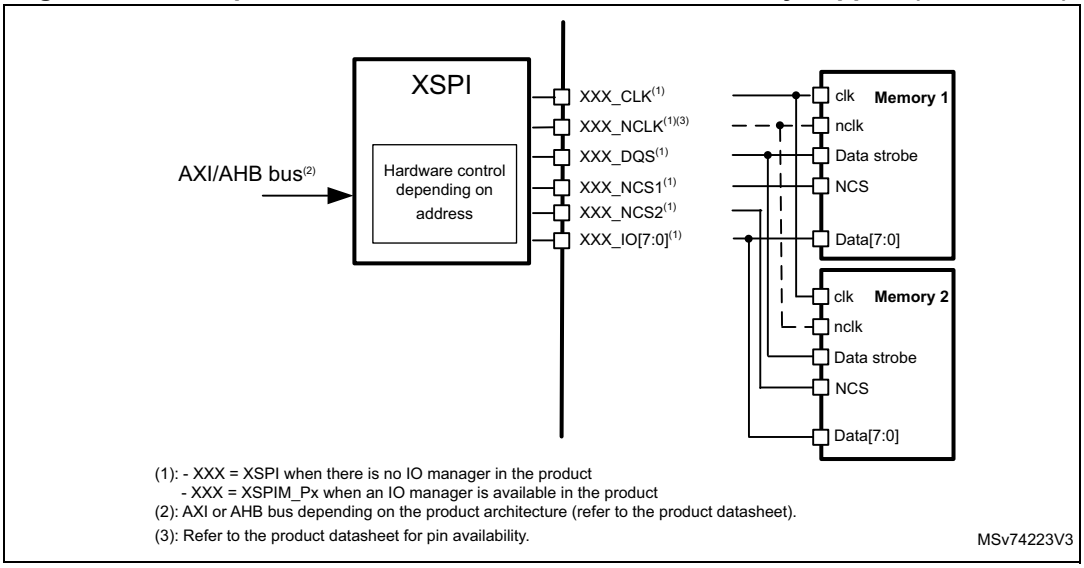
The aim is to extend by a factor two, the available external memory size. Both memories must be contiguous in the memory map. Both external memories must be of the same type and size. The size of one external memory must be set in DEVSIZE of XSPI_CR.

- When the address presented on the interconnect bus in memory-mapped mode or in XSPI_AR in indirect mode is in the range [XSPI_base_address: XSPI_base_address + $2^{DEVSIZE}$ - 1], NCS1 is generated.
- When the address presented on the interconnect bus in memory-mapped mode or in XSPI_AR in indirect mode is in the range [XSPI_base_address + $2^{DEVSIZE}$: XSPI_base_address + 2 × $2^{DEVSIZE}$ - 1], NCS2 is generated.

Note: If the access crosses the middle boundary of the external memory, CS changes automatically (no write into XSPI_AR in indirect mode, or new memory access in memory-mapped mode is needed).

The EXTENDMEM feature (set in bit 21 of XSPI_DCR1) and the Dual-memory mode (set in bit 6 of XSPI_CR) are exclusive: EXTENDMEM = 1 and DMM = 1 as this configuration is not supported.

Figure 187. Example of hardware-controller extended memory support (octal mode)



Note: It is recommended to set EXTENDMEM = 0 for write transfers targeting flash memories (HyperFlash, NOR Flash, etc.).

These memories require preliminary commands that do not include an address phase used by the controller to select the targeted memory.

This recommendation does not apply to RAM memories, as those extra commands are not needed for RAM access.

24.5 Address alignment and data number

The table below summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 215. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
	MM write				
Single, dual, quad flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.

Table 215. Address alignment cases (continued)

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Octal flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
	MM write				
Octal memory in DTR mode without WDM(6) or 8-bit HyperFLASH	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				
8-bit HyperRAM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode.
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.
7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

24.6 XSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.

Table 216. XSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

24.7 XSPI registers

24.7.1 XSPI control register (XSPI_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MSEL	FMODE[1:0]		Res.	Res.	NOPREF	CSSEL	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
	rw	rw	rw			rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTHRES[4:0]					Res.	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
			rw	rw	rw	rw	rw		rw			rw	rw	w	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **MSEL**: External memory select

This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).

0: External memory 1 selected (data exchanged over IO[3:0])

1: External memory 2 selected (data exchanged over IO[7:4])

This bit is ignored when DMM = 1 or when octal-SPI mode is selected.

Note: This bitfield can be modified only when BUSY = 0.

Bits 29:28 **FMODE[1:0]**: Functional mode

This bitfield defines the XSPI functional mode of operation.

00: Indirect-write mode

01: Indirect-read mode

10: Automatic status-polling mode (relevant in regular-command protocol only)

11: Memory-mapped mode

If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.

Note: This bitfield can be modified only when BUSY = 0.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **NOPREF**: No prefetch data
This bit disables the automatic prefetch in the external memory.
0: Automatic prefetch enabled
1: Automatic prefetch disabled

Bit 24 **CSSEL**: Chip-select selection
This bit indicates if it must copy NCS to NCS1 or NCS2.
0: NCS1 active
1: NCS2 active
Note: This bit can be modified only when BUSY = 0.

Bit 23 **PMM**: Polling match mode
This bit indicates which method must be used to determine a match during the automatic status-polling mode.
0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.
1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.
Note: This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic status-polling mode stop
This bit determines if the automatic status-polling mode is stopped after a match.
0: Automatic status-polling mode is stopped only by abort or by disabling the XSPI.
1: Automatic status-polling mode stops as soon as there is a match.
Note: This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: Timeout interrupt enable
This bit enables the timeout interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **SMIE**: Status-match interrupt enable
This bit enables the status-match interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level

This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in XSPI_SR, to be set.

00000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.

00001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.

...

11111: FTF is set if there are 32 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in indirect-read mode.

Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DMM**: Dual-memory configuration

This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity

0: Dual-memory configuration disabled

1: Dual-memory configuration enabled

Note: This bit can be modified only when BUSY = 0.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.

1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via XSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for indirect mode

1: DMA enabled for indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the XSPI.

0: XSPI disabled

1: XSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

24.7.2 XSPI device configuration register 1 (XSPI_DCR1)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	MTYP[2:0]			Res.	Res.	EXTENDMEM	DEVSIZE[4:0]				
					rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
		rw	rw	rw	rw	rw	rw					rw		rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **MTYP[2:0]**: Memory type

This bitfield indicates the type of memory to be supported.

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.

Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes.

010: Standard mode. D0/D1 ordering with DQS and CLK having the same polarity during read operation. This is the mode to use when the targeted memory does not fit with any others MTYP configuration values. This mode is typically the one targeting the APmemory 8-bit memories for instance.

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).

100: HyperBus memory mode, the protocol follows the HyperBus specification.

101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be noncacheable, or indirect-read/write modes must be used.

Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **EXTENDMEM**: Extended memory support

This bit is only relevant when one XSPI drives two same size external memories located in contiguous places in the memory map. In this case, DEVSIZE[4:0] represents the size of each external memory. When this bit is set, the XSPI automatically controls which of NCS1 or NCS2 is active.

0: NCS1 and NCS2 values depend from CSSEL bit in XSPI_CR (software controlled).

1: NCS1 and NCS2 values depend from the address of transfer (hardware controlled).

Note: It is recommended to keep EXTENDMEM = 0 for write operations targeting flash memories (such as NOR Flash and HyperFlash).

Bits 20:16 **DEVSIZE[4:0]**: Device size

This bitfield defines the size of the external device using the following formula:
Number of bytes in device = $2^{[DEVSIZE+1]}$.
DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.
In regular-command protocol, if DMM = 1 or EXTENDMEM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **CSHT[5:0]**: Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.

0x0: NCS stays high for at least 1 cycle between external device commands.

0x1: NCS stays high for at least 2 cycles between external device commands.

...

0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DLYBYP**: Delay block bypass

0: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the peripheral).

1: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.

0: CLK is not free running.

1: CLK is free running (always provided).

Note: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set. Once the bit has been set to 1, the BSY bit is set, the NCS signal remains set to 1 and the clock is sent. The only way to stop the clock is to perform an abort (which clears the BSY bit). Then, the FRCK bit must be cleared by software to allow controller transactions to take place with the external memory.

Bit 0 **CKMODE**: Clock mode 0

This bit is read-only and is read as zero.

24.7.3 XSPI device configuration register 2 (XSPI_DCR2)

Address offset: 0x00C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size

This bitfield indicates the wrap size to which the memory is configured. For memories which have a separate command for wrapped instructions, this bitfield indicates the wrap-size associated with the command held in the XSPI_WPIR register.

000: Wrapped reads are not supported by the memory.

010: External memory supports wrap size of 16 bytes.

011: External memory supports wrap size of 32 bytes.

100: External memory supports wrap size of 64 bytes.

101: External memory supports wrap size of 128 bytes.

Other: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as XSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the XSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50 %. The clock signal remains low one cycle longer than it stays high.

24.7.4 XSPI device configuration register 3 (XSPI_DCR3)

Address offset: 0x010

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary

This bitfield enables the transaction boundary feature. When active, a minimum value of 3 is recommended. The NCS is released on each boundary of $2^{CSBOUND}$ bytes.

0: NCS boundary disabled

Other: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:0 Reserved, must be kept at reset value.

24.7.5 XSPI device configuration register 4 (XSPI_DCR4)

Address offset: 0x014

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFRESH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFRESH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REFRESH[31:0]**: Refresh rate

This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual- or quad-SPI mode, because the byte transmission must be completed.

0: Refresh disabled

Other: Maximum communication length is set to REFRESH + 1 clock cycles.

Note: REFRESH count is based on the memory clock period: if XSPI_DCR2 PRESCALER bitfield is changed, the REFRESH field must be updated accordingly.

24.7.6 XSPI status register (XSPI_SR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
		r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full.

In automatic status-polling mode, FLEVEL is zero.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **BUSY**: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (XSPI_PSMAR).

It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.

It is cleared automatically as soon as the threshold condition is no longer true.

In automatic status-polling mode, this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode.

It is cleared by writing 1 to CTEF.

24.7.7 XSPI flag clear register (XSPI_FCR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
											w	w		w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag
Writing 1 clears the TOF flag in the XSPI_SR register.

Bit 3 **CSMF**: Clear status match flag
Writing 1 clears the SMF flag in the XSPI_SR register.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag
Writing 1 clears the TCF flag in the XSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in the XSPI_SR register.

24.7.8 XSPI data length register (XSPI_DLR)

Address offset: 0x040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.

All 1's in indirect mode means undefined length, where XSPI continues until the end of the memory, as defined by DEVSIZE.

0x0000_0000: 1 byte is to be transferred.

0x0000_0001: 2 bytes are to be transferred.

0x0000_0002: 3 bytes are to be transferred.

0x0000_0003: 4 bytes are to be transferred.

...

0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.

0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.

0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.

DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.

This bitfield has no effect in memory-mapped mode.

24.7.9 XSPI address register (XSPI_AR)

Address offset: 0x048

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0 and FMODE ≠ 11.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be sent to the external device. In dual-memory configuration, AR[0] is forced to 0.

Caution: Some memory specifications consider that each address corresponds to a 16-bit value. XSPI considers that each address corresponds to an 8-bit value. So the software needs to multiply the address by two when accessing the memory registers.

24.7.10 XSPI data register (XSPI_DR)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes.

Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

24.7.11 XSPI polling status mask register (XSPI_PSMKR)

Address offset: 0x080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.

24.7.12 XSPI polling status match register (XSPI_PSMAR)

Address offset: 0x088

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MATCH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MATCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MATCH[31:0]**: Status match
Value to be compared with the masked status register to get a match

24.7.13 XSPI polling interval register (XSPI_PIR)

Address offset: 0x090

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval
Number of CLK cycles between a read during the automatic status-polling phases

24.7.14 XSPI communication configuration register (XSPI_CCR)

Address offset: 0x100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate- byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Other: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

24.7.15 XSPI timing configuration register (XSPI_TCR)

Address offset: 0x108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle
0: No delay hold
1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

24.7.16 XSPI instruction register (XSPI_IR)

Address offset: 0x110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

24.7.17 XSPI alternate bytes register (XSPI_ABR)

Address offset: 0x120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address.

24.7.18 XSPI low-power timeout register (XSPI_LPTR)

Address offset: 0x130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the XSPI prefetches the subsequent bytes and hold them in the FIFO.

This bitfield indicates how many CLK cycles the XSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

24.7.19 XSPI wrap communication configuration register (XSPI_WPCCR)

Address offset: 0x140

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable

This bit enables the data strobe management.

0: DQS disabled

1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: no alternate bytes
001: alternate bytes on a single line
010: alternate bytes on two lines
011: alternate bytes on four lines
100: alternate bytes on eight lines
Other: reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

24.7.20 XSPI wrap timing configuration register (XSPI_WPTCR)

Address offset: 0x148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1).

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle
Add a quarter cycle delay on the outputs in DTR communication to match hold requirement.
0: No quarter cycle delay
1: 1/4 cycle delay inserted

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

24.7.21 XSPI wrap instruction register (XSPI_WPIR)

Address offset: 0x150

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

24.7.22 XSPI wrap alternate bytes register (XSPI_WPABR)

Address offset: 0x160

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address

24.7.23 XSPI write communication configuration register (XSPI_WCCR)

Address offset: 0x180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Other: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate-bytes phase.
0: DTR mode disabled for alternate-bytes phase
1: DTR mode enabled for alternate-bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Other: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Other: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Other: Reserved

24.7.24 XSPI write timing configuration register (XSPI_WTCR)

Address offset: 0x188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency. In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

24.7.25 XSPI write instruction register (XSPI_WIR)

Address offset: 0x190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

24.7.26 XSPI write alternate bytes register (XSPI_WABR)

Address offset: 0x1A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address

24.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR)

Address offset: 0x200

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TRWR[7:0]:** Read-write minimum recovery time

Device read-to-write/write-to-read minimum recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7:0]**: Access time

Device access time according to the memory latency, expressed in number of communication clock cycles

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **WZL:** Write zero latency

This bit enables zero latency on write operations.

0: Latency on write accesses

1: No latency on write accesses

Bit 0 **LM:** Latency mode

This bit selects the latency mode.

0: Variable initial latency

1: Fixed latency

Note: This bit must be set when using the dual-octal HyperRAM configuration. It must be kept cleared for HyperFLASH memory.

24.7.28 XSPI register map

Table 217. XSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	XSPI_CR	Res.	MSEL	FMDOE[1:0]		Res.	Res.	NOPREF	CSSEL	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]					Res.	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
	Reset value		0	0	0			0	0	0	0		0	0	0	0	0				0	0	0	0	0		0			0	0	0	0
0x004	Reserved																																
0x008	XSPI_DCR1	Res					MTYP [2:0]			Res.	Res.	EXTENDMEM	DEVSIZE[4:0]					Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value						0	0	0			0	0	0	0	0	0			0	0	0	0	0	0					0		0	0
0x00C	XSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x010	XSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0	0	0	0																
0x014	XSPI_DCR4	REFRESH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved																																
0x020	XSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x024	XSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																												0	0		0	0
0x028-0x03C	Reserved																																
0x040	XSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	Reserved	Reserved																															

Table 217. XSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	XSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	Reserved																																
0x050	XSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054-0x07C	Reserved																																
0x080	XSPI_PSMKR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	Reserved																																
0x088	XSPI_PSMAR	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	Reserved																																
0x090	XSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094-0x0FC	Reserved																																
0x100	XSPI_CCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x104	Reserved																																
0x108	XSPI_TCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x10C	Reserved																																
0x110	XSPI_IR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114-0x11C	Reserved																																
0x120	XSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124-0x12C	Reserved																																
0x130	XSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134-0x13C	Reserved																																

Table 217. XSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x140	XSPI_WPCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x144	Reserved																																
0x148	XSPI_WPTCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x14C	Reserved																																
0x150	XSPI_WPIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154-0x15C	Reserved																																
0x160	XSPI_WPABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x164-0x17C	Reserved																																
0x180	XSPI_WCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x184	Reserved																																
0x188	XSPI_WTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value																												0	0	0	0	0
0x18C	Reserved																																
0x190	XSPI_WIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94-0x019C	Reserved																																
0x1A0	XSPI_WABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A4-0x1FC	Reserved																																
0x200	XSPI_HLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]								TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

25 Delay block (DLYB)

25.1 DLYB introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an XSPI interface.

The delay is voltage- and temperature-dependent, that may require the application to re-configure and recenter the output clock phase with the receive data.

25.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

25.3 DLYB implementation

Table 218. STM32G56x/55x/54x/53x features

DLYB associated peripheral	DLYBXS1
XSPI1	X

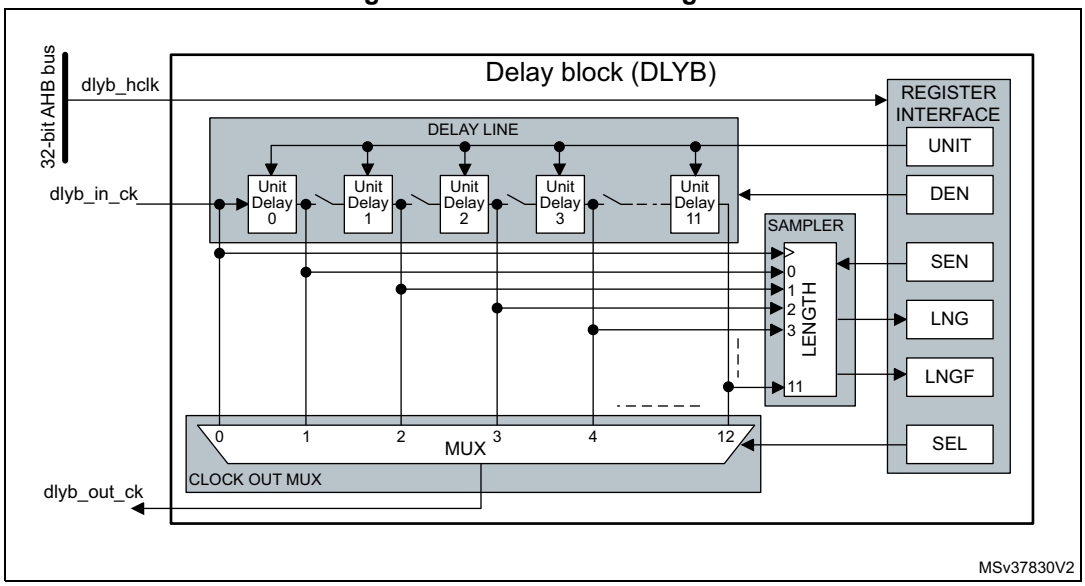
25.4 DLYB functional description

25.4.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 188. DLYB block diagram



25.4.2 DLYB pins and internal signals

Table 219 lists the DLYB internal signals.

Table 219. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

Table 220. DLYB interconnection

Signal name	Source/destination
	XSPI1
dlyb_in_ck	DQS
dlyb_out_ck	DQS delayed for data sampling

25.4.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 221. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don't care	Don't care	Don't care	Don't care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don't care	Don't care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.

25.4.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

25.4.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

XSPI use case:

The delay block is used in conjunction with XSPI interface to allow shifting the input data sampling signal. This sampling signal can be the feedback clock or the data strobe (DQS) signal, which is delivered by certain type of devices. Note that in case DQS is used, the calibration procedure must be performed beforehand with a free running clock, as DQS is a discontinuous signal.

In case of SDR (single data rate) mode the user must typically shift the sampling signal by half period, so that the sampling edges are positioned in the middle of the valid data phase.

In case of DDR (dual data rate) mode, for which data are transitioning at start and middle of period, typical value must be close to N/4, once the calibration is completed.

In case of high frequencies and tight timing constraints, the delay setting granularity (10) might be too coarse. Since in most cases it is not necessary to have a possible delay value covering the whole sampling clock period, the "Unit" value can be overridden by application in order to improve the accuracy of the sampling edge position (example: providing a twice as small “Unit” gives twice better timing accuracy. The counterpart being that the maximum possible delay is divided by 2).

25.5 DLYB registers

All registers can be accessed in word, half-word and byte access.

25.5.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SEN:** Sampler length enable bit

0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.

1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit

0: DLYB disabled.

1: DLYB enabled.

25.5.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LNGF	Res.	Res.	Res.	LNG[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **LNGF:** Length valid flag

This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.

0: Length value in LNG is not valid.

1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value

These bits reflect the 12 unit delay values sampled at the rising edge of the input clock. The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay of a unit delay cell.
These bits can only be written when SEN = 1.
Unit delay = initial delay + UNIT[6:0] x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]:** Phase for the output clock.
These bits can only be written when SEN = 1.
Output clock phase = input clock + SEL[3:0] x unit delay

25.5.3 DLYB register map

Table 222. DLYB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DLYB_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
	Reset value																															0	0
0x004	DLYB_CFGR	LNGF	Res.	Res.	Res.	LNG[11:0]												Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

26 Analog-to-digital converters (ADC)

26.1 ADC introduction

This section describes the implementation of up to 3 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).
- ADC3 is controlled independently.

Each ADC consists of one 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan, or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned (default configuration) 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

26.2 ADC main features

- High-performance features
  - Up to 3 ADCs, out of which two of them can operate in dual mode:
    ADC1 is connected to 15 external channels + 5 internal channels
    ADC2 is connected to 17 external channels + 4 internal channels
    ADC3 is connected to 16 external channels + 4 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Management of single-ended, pseudodifferential, and differential inputs (programmable per channels)
  - AHB slave bus interface to allow fast data handling
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Fixed latency for a trigger to start of sampling
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Four dedicated data registers for the injected channels

- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (autodelayed mode)
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024
  - Programmable data right and left shifts
  - Dedicated accumulator for each regular and injected channel
  - Parallel oversampling mode for injected channels (access to the intermediate oversampling result)
- Data preconditioning
  - Gain compensation
  - Offset compensation
- Analog input channels
  - External analog inputs (per ADC): up to 17 GPIO pads
  - 3 channels for the internal reference voltage (V_{REFINT})
  - 1 channel for the internal temperature sensor (V_{SENSE})
  - 3 channels for monitoring V_{DDCORE} internal voltage
  - Connection to OPAMP internal channels
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timer events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC, with filters
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 189 shows the block diagram of one ADC.

26.3 ADC implementation

Table 223. ADC features

ADC modes/features	ADC1, ADC2	ADC3[1]
Resolution	12 bits	
Maximum sampling speed	5 Msps (12-bit resolution)	
Dual mode operation	X	-
Offset calibration	X	
Single-end input	X	
Differential input	X	
Injected channel conversion	X	
Oversampling	up to x1024	
Data register	32 bits	
DMA support	X	
Parallel data output to MDF and ADF	-	
Offset compensation	X	
Gain compensation	X	
Number of analog watchdog	3	

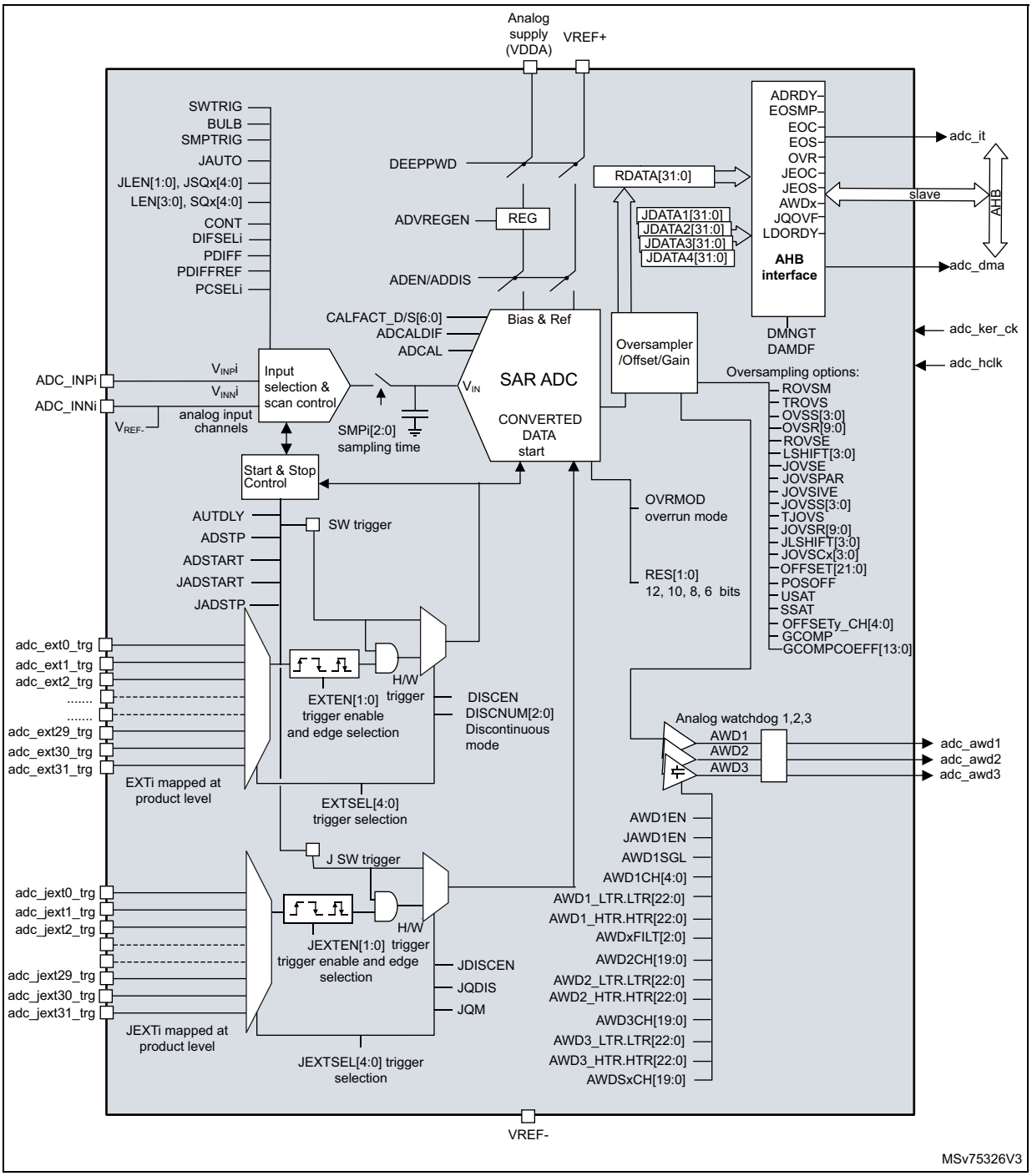
1. ADC3 is only available on STM32G55x/56x devices.

26.4 ADC functional description

26.4.1 ADC block diagram

Figure 189 shows the ADC block diagram and *Table 224* gives the ADC pin description.

Figure 189. ADC block diagram



26.4.2 ADC pins and internal signals

Table 224. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
VREF-	Input, analog reference negative	The lower/negative reference voltage for the ADC.
ADCx_INNi/INPi	Negative/positive external analog input signals	20 negative/positive external analog input channels (refer to *Section 26.4.4: ADC connectivity* for details).

Table 225. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{INP}i$	Positive analog input channels	Positive internal analog input channels connected either to ADCx_INP*i* external channels or to internal channels.
$V_{INN}i$	Negative analog input channels	Negative internal analog input channels connected either to ADCx_INN*i* external channels or to internal channels.
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.These inputs are shared between the ADC master and the ADC slave.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.
adc_awdy	Output	Internal analog watchdog output signal connected to on-chip timers. (y = Analog watchdog number 1,2,3).
adc_ker_ck	Input	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request

Table 226. ADC1/2/3 interconnection

Signal name	Source/destination
ADC1 V_{INP}[13]	opamp1_int(1)
ADC1 V_{INP}[16]	V_{SENSE}
ADC1 V_{INP}[17]	V_{DDCORE}
ADC1 V_{INP}[18]	V_{REFINT}
ADC1 V_{INP}[19]	opamp2_int(1)

Table 226. ADC1/2/3 interconnection (continued)

Signal name	Source/destination
ADC2 V_{INP}[10]	V_{DDCORE}
ADC2 V_{INP}[16]	opamp2_int(1)
ADC2 V_{INP}[18]	V_{REFINT}
ADC2 V_{INP}[19]	opamp3_int(1)
ADC3 V_{INN}[9]	V_{DDCORE}
ADC3 V_{INP}[10]	V_{DDCORE}
ADC3 V_{INP}[13]	opamp3_int
ADC3 V_{INP}[18]	V_{REFINT}
ADC3 V_{INP}[19]	opamp1_int
adc1_ext_trg0, adc2_ext_trg0	exti11
adc1_ext_trg1, adc2_ext_trg1	tim1_oc1
adc1_ext_trg2, adc2_ext_trg2	tim1_oc2
adc1_ext_trg3, adc2_ext_trg3	tim1_oc3
adc1_ext_trg4, adc2_ext_trg4	tim1_trgo
adc1_ext_trg5, adc2_ext_trg5	tim1_trgo2
adc1_ext_trg6, adc2_ext_trg6	tim2_oc2
adc1_ext_trg7, adc2_ext_trg7	tim2_trgo
adc1_ext_trg8, adc2_ext_trg8	tim3_oc4 / tm4_oc4(1)
adc1_ext_trg9, adc2_ext_trg9	tim3_trgo / tim4_trgo(1)(2)
adc1_ext_trg10, adc2_ext_trg10	tim6_trgo
adc1_ext_trg11, adc2_ext_trg11	tim8_trgo
adc1_ext_trg12, adc2_ext_trg12	tim8_trgo2
adc1_ext_trg13, adc2_ext_trg13	tim9_trgo
adc1_ext_trg14, adc2_ext_trg14	tim15_trgo
adc1_ext_trg15, adc2_ext_trg15	tim20_oc1(1)
adc1_ext_trg16, adc2_ext_trg16	tim20_oc2(1)
adc1_ext_trg17, adc2_ext_trg17	tim20_trgo(1)
adc1_ext_trg18, adc2_ext_trg18	tim20_trgo2(1)
adc1_ext_trg19, adc2_ext_trg19	tim26_trgo
adc1_ext_trg20, adc2_ext_trg20	lptim1_ch1
adc1_ext_trg21, adc2_ext_trg21	play1_out11(1)
adc1_ext_trg22, adc2_ext_trg22	hrtim_adc_trg1
adc1_ext_trg23, adc2_ext_trg23	hrtim_adc_trg2
adc1_ext_trg24, adc2_ext_trg24	hrtim_adc_trg3
adc1_ext_trg25, adc2_ext_trg25	hrtim_adc_trg4

Table 226. ADC1/2/3 interconnection (continued)

Signal name	Source/destination
adc1_ext_trg26, adc2_ext_trg26	hrtim_adc_trg5
adc1_ext_trg27, adc2_ext_trg27	hrtim_adc_trg6
adc1_ext_trg28, adc2_ext_trg28	hrtim_adc_trg7
adc1_ext_trg29, adc2_ext_trg29	hrtim_adc_trg8
adc1_ext_trg30, adc2_ext_trg30	hrtim_adc_trg9
adc1_ext_trg31, adc2_ext_trg31	hrtim_adc_trg10
adc1_jext_trg0, adc2_jext_trg0	exti15
adc1_jext_trg1, adc2_jext_trg1	tim1_oc4
adc1_jext_trg2, adc2_jext_trg2	tim1_trgo
adc1_jext_trg3, adc2_jext_trg3	tim1_trgo2
adc1_jext_trg4, adc2_jext_trg4	tim2_oc1
adc1_jext_trg5, adc2_jext_trg5	tim2_trgo
adc1_jext_trg6, adc2_jext_trg6	tim3_oc1(1)
adc1_jext_trg7, adc2_jext_trg7	tim3_oc3(1)
adc1_jext_trg8, adc2_jext_trg8	tim3_oc4(1)
adc1_jext_trg9, adc2_jext_trg9	tim3_trgo / tim4_trgo(1)(2)
adc1_jext_trg10, adc2_jext_trg10	tim7_trgo
adc1_jext_trg11, adc2_jext_trg11	tim8_oc4
adc1_jext_trg12, adc2_jext_trg12	tim8_trgo
adc1_jext_trg13, adc2_jext_trg13	tim8_trgo2
adc1_jext_trg14, adc2_jext_trg14	tim12_trgo
adc1_jext_trg15, adc2_jext_trg15	tim15_trgo
adc1_jext_trg16, adc2_jext_trg16	tim16_oc1
adc1_jext_trg17, adc2_jext_trg17	tim20_trgo(1)
adc1_jext_trg18, adc2_jext_trg18	tim20_trgo2(1)
adc1_jext_trg19, adc2_jext_trg19	tim27_trgo
adc1_jext_trg20, adc2_jext_trg20	lptim1_ch2
adc1_jext_trg21, adc2_jext_trg21	play1_out13(1)
adc1_jext_trg22, adc2_jext_trg22	hrtim_adc_trg1
adc1_jext_trg23, adc2_jext_trg23	hrtim_adc_trg2
adc1_jext_trg24, adc2_jext_trg24	hrtim_adc_trg3
adc1_jext_trg25, adc2_jext_trg25	hrtim_adc_trg4
adc1_jext_trg26, adc2_jext_trg26	hrtim_adc_trg5
adc1_jext_trg27, adc2_jext_trg27	hrtim_adc_trg6
adc1_jext_trg28, adc2_jext_trg28	hrtim_adc_trg7

Table 226. ADC1/2/3 interconnection (continued)

Signal name	Source/destination
adc1_jext_trg29, adc2_jext_trg29	hrtim_adc_trg8
adc1_jext_trg30, adc2_jext_trg30	hrtim_adc_trg9
adc1_jext_trg31, adc2_jext_trg31	hrtim_adc_trg10
adc3_ext_trg0	exti2
adc3_ext_trg1	tim1_oc3
adc3_ext_trg2	tim1_trgo
adc3_ext_trg3	tim1_trgo2
adc3_ext_trg4	tim2_oc1
adc3_ext_trg5	tim2_oc3
adc3_ext_trg6	tim2_trgo
adc3_ext_trg7	tim3_oc1
adc3_ext_trg8	tim3_oc2 / tim4_oc1(2)
adc3_ext_trg9	tim3_trgo / tim4_trgo(2)
adc3_ext_trg10	tim6_trgo
adc3_ext_trg11	tim8_oc1
adc3_ext_trg12	tim8_trgo
adc3_ext_trg13	tim8_trgo2
adc3_ext_trg14	tim9_trgo
adc3_ext_trg15	tim15_trgo
adc3_ext_trg16	tim20_oc1
adc3_ext_trg17	tim20_trgo
adc3_ext_trg18	tim20_trgo2
adc3_ext_trg19	tim26_trgo
adc3_ext_trg20	lptim1_ch1
adc3_ext_trg21	play1_out12
adc3_ext_trg22	hrtim_adc_trg1
adc3_ext_trg23	hrtim_adc_trg2
adc3_ext_trg24	hrtim_adc_trg3
adc3_ext_trg25	hrtim_adc_trg4
adc3_ext_trg26	hrtim_adc_trg5
adc3_ext_trg27	hrtim_adc_trg6
adc3_ext_trg28	hrtim_adc_trg7
adc3_ext_trg29	hrtim_adc_trg8
adc3_ext_trg30	hrtim_adc_trg9
adc3_ext_trg31	hrtim_adc_trg10

Table 226. ADC1/2/3 interconnection (continued)

Signal name	Source/destination
adc3_jext_trg0	exti3
adc3_jext_trg1	tim1_oc3
adc3_jext_trg2	tim1_oc4
adc3_jext_trg3	tim1_trgo
adc3_jext_trg4	tim1_trgo2
adc3_jext_trg5	tim2_trgo
adc3_jext_trg6	tim3_trgo
adc3_jext_trg7	tim7_trgo
adc3_jext_trg8	tim8_oc2 / tim4_oc3(2)
adc3_jext_trg9	tim8_oc4 / tim4_trgo(2)
adc3_jext_trg10	tim8_trgo
adc3_jext_trg11	tim8_trgo2
adc3_jext_trg12	tim12_trgo
adc3_jext_trg13	tim15_trgo
adc3_jext_trg14	tim16_oc1
adc3_jext_trg15	tim17_oc1
adc3_jext_trg16	tim20_oc2
adc3_jext_trg17	tim20_trgo
adc3_jext_trg18	tim20_trgo2
adc3_jext_trg19	tim27_trgo
adc3_jext_trg20	lptim1_ch2
adc3_jext_trg21	play1_out14
adc3_jext_trg22	hrtim_adc_trig1
adc3_jext_trg23	hrtim_adc_trig2
adc3_jext_trg24	hrtim_adc_trig3
adc3_jext_trg25	hrtim_adc_trig4
adc3_jext_trg26	hrtim_adc_trig5
adc3_jext_trg27	hrtim_adc_trig6
adc3_jext_trg28	hrtim_adc_trig7
adc3_jext_trg29	hrtim_adc_trig8
adc3_jext_trg30	hrtim_adc_trig9
adc3_jext_trg31	hrtim_adc_trig10

1. This connection is only available on STM32G55x/56x devices.
2. This input can be selected through the option register.

26.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC kernel clock is independent from the AHB bus clock that is used to access ADC registers.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 190: ADC clock scheme*). This selection is done in the RCC (refer to section *Reset and clock control (RCC)* for more information):

1. The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous from the AHB clock.
2. The ADC clock can be derived from AHB clock (selected in RCC).

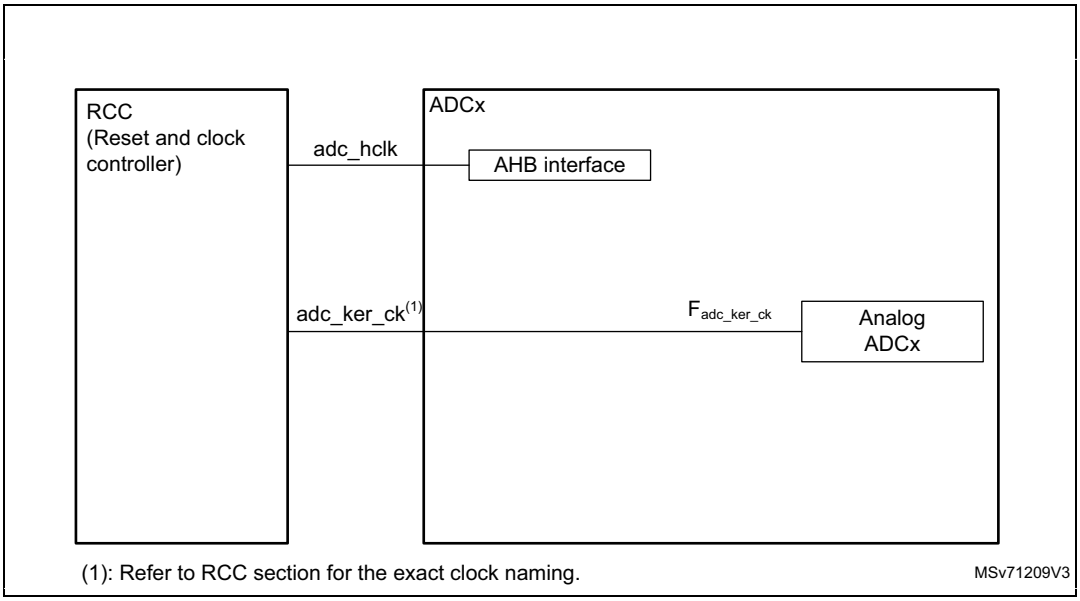
Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected.

Option **2** corresponds to a pseudosynchronous clock. This can be useful when the ADC is triggered by a timer and the application requires that the ADC is accurately triggered without any uncertainty (otherwise, an uncertainty of the trigger instant time is added by the resynchronizations between the two clock domains). This accurate trigger is supported only by trgo or trgo2 timer triggers.

For further details on the synchronization between ADC and timers, refer to section *ADC synchronization* in each timer section, and to section *Peripheral clock distribution* in the RCC section.

The clock is configured through the RCC and must be compliant with the operating frequency specified in the device datasheet.

Figure 190. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

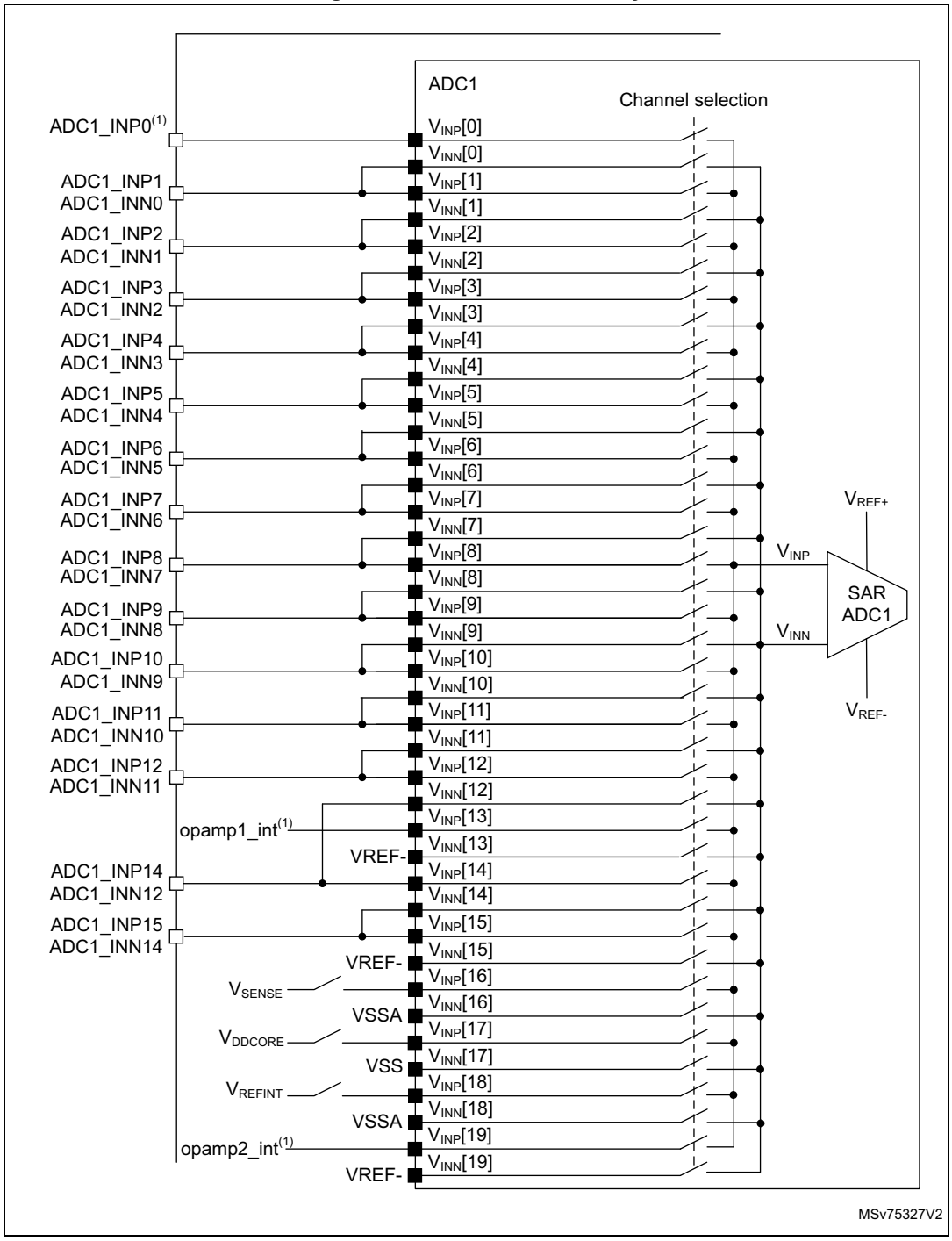
There are no constraints to respect for the ratio between the ADC clock and the AHB clock. However, the ratio must be carefully chosen to avoid any overrun especially if the clock AHB is much slower than the ADC clock.

When adc_hclk operates at a higher frequency than adc_ker_ck, writing to the AHB register does not immediately update the ADC configuration due to clock domain crossing delays. To guarantee that the configuration is properly applied within the adc_ker_ck domain, it is necessary to wait for four adc_ker_ck clock cycles after the AHB register update.

26.4.4 ADC connectivity

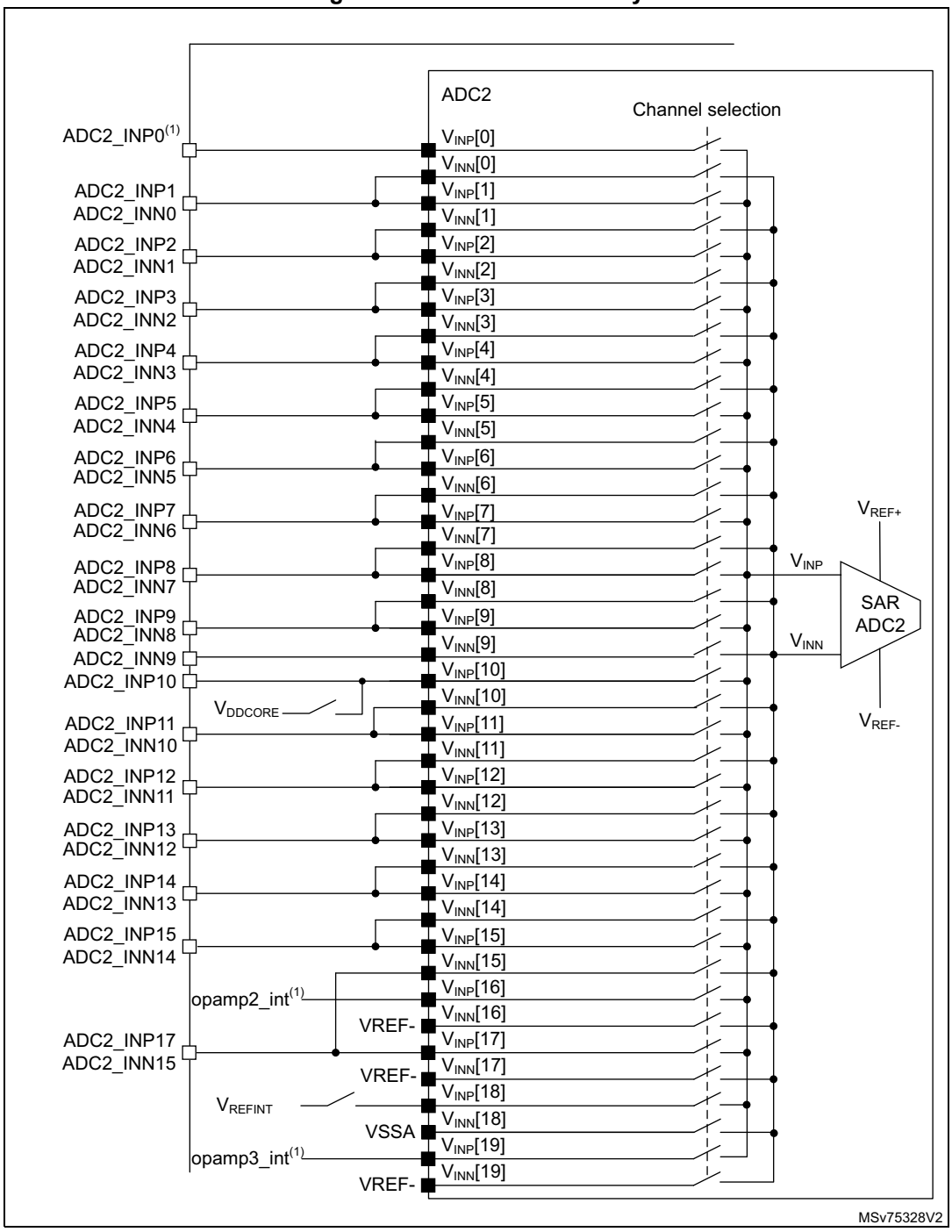
ADC inputs are connected to external channels as well as internal sources as described below.

Figure 191. ADC1 connectivity



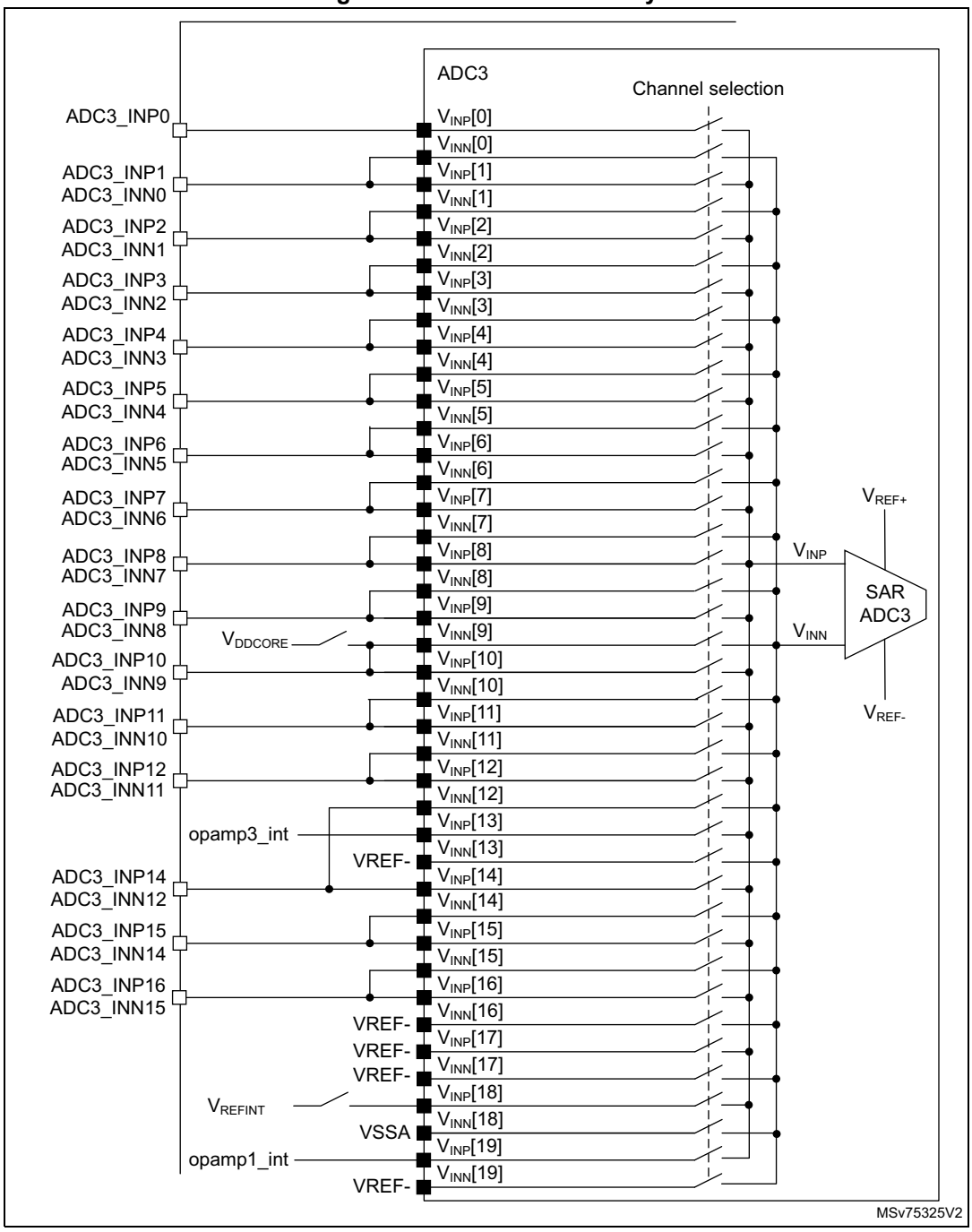
1. This connection is only available on STM32G55x/56x devices.

Figure 192. ADC2 connectivity



1. This connection is only available on STM32G55x/56x devices.

Figure 193. ADC3 connectivity



26.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and it can generate AHB error in case of wrong HSIZE.

26.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in deep-power-down mode where its supply voltage is internally switched off to reduce the leakage currents (the reset state of the DEEPPWD bit is 1 in the ADC_CR register).

To activate the ADC, first exit the Deep-power-down mode by clearing the DEEPPWD bit.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the ADVREGEN bit of the ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG_STUP}$) before launching a calibration or enabling the ADC. This delay must be implemented by software. For the startup time of the ADC voltage regulator, refer to the device datasheet for $T_{ADCVREG_STUP}$ parameter.

When ADC operations are complete, the ADC can be disabled by clearing the ADEN bit of the ADC_CR register. Power can then be saved by disabling the ADC voltage regulator (by clearing ADVREGEN).

Additional power can be saved by entering ADC Deep-power-down mode again (by setting DEEPPWD of ADC_CR register). This is particularly interesting before entering Stop mode.

Note: Setting DEEPPWD automatically disables the ADC voltage regulator and the ADVREGEN bit is automatically cleared.

When the internal voltage regulator is disabled (ADVREGEN cleared), the internal analog calibration is kept.

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or reapply the calibration factor, which was previously saved (refer to *Section Calibration*).

26.4.7 Single-ended and differential input channels

ADC channels can be configured to be either single-ended inputs or differential inputs by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be programmed when no ADC conversion is ongoing. Note that the DIFSEL[i] bits corresponding to single-ended channels must be programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel "*i*" is the difference between the external voltage $V_{INP[i]}$ and V_{REF-}.

In differential input mode, the analog voltage to be converted for channel "*i*" is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

In differential mode, the output data are unsigned data. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals V_{REF+} and the output data is 0x000 (12-bit resolution mode). When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the output data is 0xFFF.

$$\text{Converted value} = \frac{\text{ADC_Full_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When the ADC is configured in differential mode, the input signals are supposed to be differential (the common-mode voltage must be fixed at V_{REF+}/2 voltage).

Internal channels are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to *Section 26.4.4: ADC connectivity*.

Caution: When configuring channel "*i*" in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in single-ended or differential mode, and must never be configured to be converted. Some channels are shared between ADCs: this can make the channel on the other ADCs unusable. The only exception is interleaved mode for ADC master and the slave.

26.4.8 Pseudodifferential mode

The ADC can operate in pseudodifferential mode instead of differential modes when DIFSELx = 1. In this mode, differential inputs are supported but the negative input $V_{INN[i]}$ is fixed to V_{REF-} or to V_{REF+}/2 voltage.

The pseudodifferential mode is selected by setting the PDIFF bit of the ADC_CFGR2 register. This changes the behavior of the differential channel selected via the ADC_DIFSEL register. In addition, the negative input reference voltage can be selected between V_{REF-} and V_{REF+}/2 through the PDIFFREF bit of the ADC_CFGR2 register.

When the ADC is configured in pseudodifferential mode:

- When $V_{INP[i]}$ equals 0 V, the output data is 0x000.
- When $V_{INP[i]}$ equals V_{REF+}, the output data is 0xFFF,

Note: The behavior is the same as in single-ended mode except that $V_{INN[i]}$ is fixed to 0 V or V_{REF+}/2.

The pseudodifferential mode can be used to reject the common mode noise that exists between the ADC shunt resistor bottom voltage (ground) and top voltage (voltage to be converted by the ADC).

The pseudodifferential mode does not affect single-end channel conversions.

The pseudodifferential and standard differential modes cannot be used simultaneously in the same scan conversion sequence. If pseudodifferential or differential conversion is required, stop and disable the ADC to change the PDIFF setting.

26.4.9 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

Each ADC provides an automatic calibration procedure that drives all the calibration sequences including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor, which is 7-bit wide and is applied internally to the ADC until

the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Clear ADCALDIF before launching a calibration that is applied for single-ended input conversions.
- Set ADCALDIF before launching a calibration that is applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). The ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of the ADC_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before reenabling the ADC.

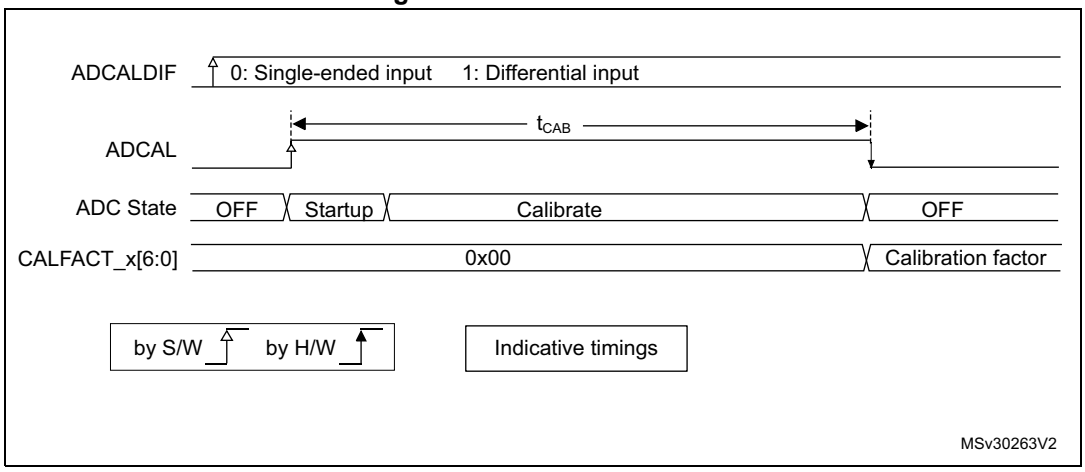
The internal analog calibration is lost each time the power of the ADC is removed (for example, when the product enters Standby). In this case, to avoid spending time recalibrating the ADC, it is possible to rewrite the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when the V_{REF+} voltage changed more than 10% (see *Section 26.4.35: Monitoring the internal voltage reference*).

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by clearing ADCALDIF (single-ended input) or by setting ADCALDIF (differential input).
4. Set ADCAL.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from the ADC_CALFACT register.

Figure 194. ADC calibration

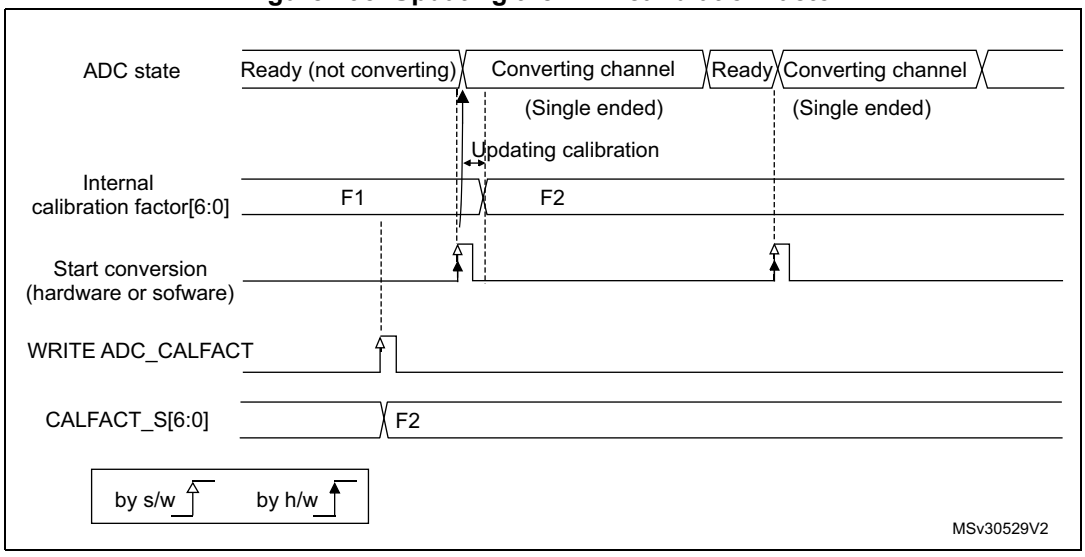


Software procedure to reinject a calibration factor into the ADC

1. Ensure ADSTART = 0 and JADSTART = 0 (no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Note: In pseudodifferential mode, only the CALFACT_D calibration factor is used.

Figure 195. Updating the ADC calibration factor

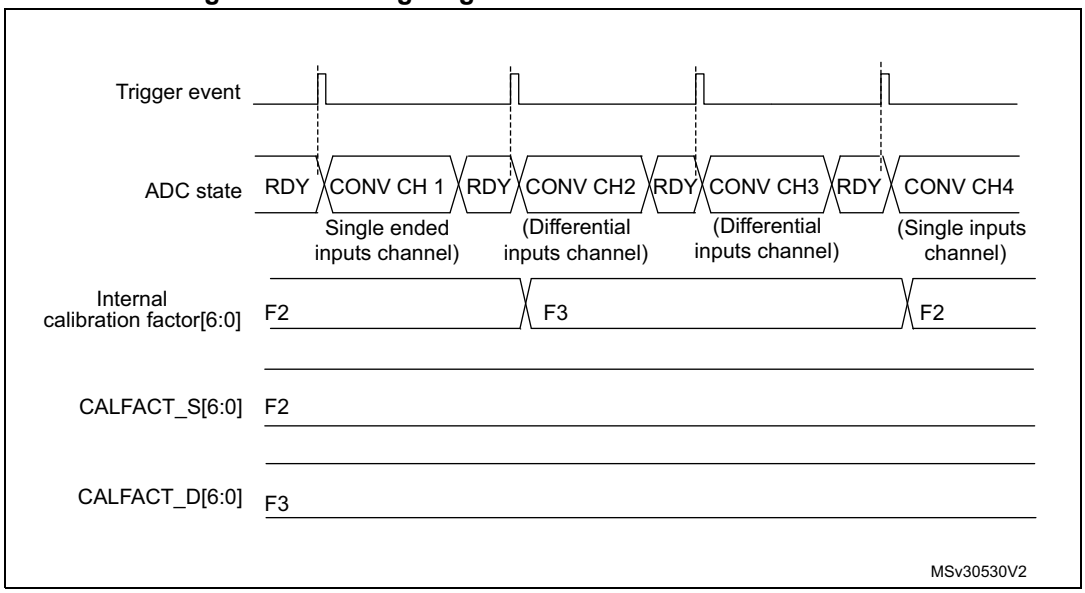


Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential input channel (and vice versa), the calibration is automatically injected into the analog ADC.

Figure 196. Mixing single-ended and differential channels



Note: The offset calibration in pseudodifferential mode is controlled by the CALFACF_D register.

26.4.10 ADC on-off control (ADEN, ADDIS, ADRDY)

First, follow the procedure described in *Section 26.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD is cleared and ADVREGEN is set, the ADC can be enabled. It requires a t_{STAB} stabilization time before starting converting accurately (see *Figure 197*).

Two control bits enable or disable the ADC:

- Set ADEN to enable the ADC. The ADRDY flag is set once the ADC is ready for operation.
- Set ADDIS to disable the ADC. ADEN and ADDIS are automatically deasserted by hardware as soon as the analog ADC is effectively disabled.

Regular conversions can then start either by setting ADSTART (refer to *Section 26.4.21: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs if regular triggers are enabled.

Injected conversions start by setting JADSTART or when an external injected trigger event occurs, if injected triggers are enabled.

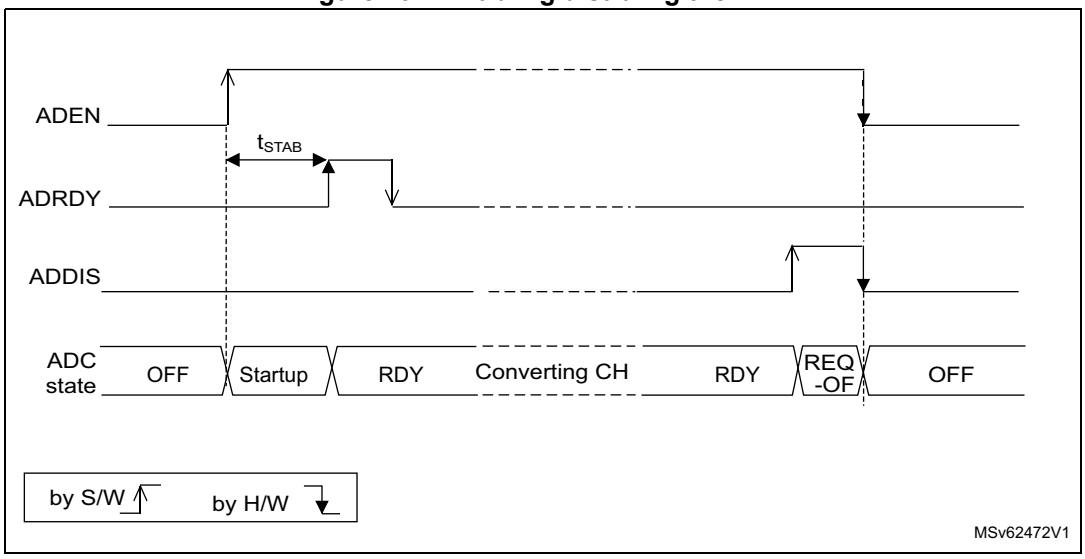
Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by programming it to 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done by using the associated interrupt (ADRDYIE must be set).
4. Clear the ADRDY bit in the ADC_ISR register by programming it to 1 (optional).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop all regular and injected ongoing conversions by setting ADSTP and JADSTP. Then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait for ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 197. Enabling/disabling the ADC



26.4.11 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC section) and configure the ADEN bit in the ADC_CR register, only if the ADC is disabled (ADEN must be cleared).

The software can modify the PDIFF and PDIFFREF of the ADC_CFGR2 only when ADEN is cleared.

The software can write the ADSTART, JADSTART and ADDIS control bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

The following constraints apply to all the other control bits of the ADC_CFGRx, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if no regular conversions are ongoing (ADSTART must be cleared).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if no injected conversions are ongoing (JADSTART must be cleared).
- ADC_TRy register thresholds can be modified when an analog-to-digital conversion is ongoing (refer to *Section 26.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)* for details).
- To modify the all above control bits when the software trigger function is used, an ADSTP or JADSTP must be issued to make sure all activities are stopped, even if ADSTART and JADSTART are cleared.

The software is allowed to write the ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and if there is no pending request to disable the ADC. In addition, the software can write the ADC_JSQR register, when injected queue are disabled by JQDIS = 1, only if JADSTART is cleared and no injected conversion is ongoing. Refer to *Section 26.6.14: ADC injected sequence register (ADC_JSQR)* for additional details.

Note: Not all forbidden write accesses to ADC control bits are protected by hardware. In some cases, the ADC state may become unknown. To recover from this situation, the ADC must be disabled (ADEN = 0 as well as all the bits of the ADC_CR register).

26.4.12 Channel selection (ADC_SQRy, ADC_JSQR)

The ADC features up to 20 multiplexed channels per ADC, out of which:

- Up to 17 analog inputs coming from GPIO pads (ADCx_INN/INP[i]). Depending on the device, not all of them are available on GPIO pads.
- The ADC is connected to up to 6 internal analog inputs:
  - internal reference voltage (V_{REFINT})
  - Internal temperature sensor (V_{SENSE})
  - V_{DDCORE} voltage
  - OPAMP internal channel

  To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming VREFEN, TSEN in the ADCC_CCR registers, or by enabling the corresponding bit in the *ADC option register (ADC_OR)*.

Refer to *ADC interconnection* tables in *Section 26.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to ADC pins.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_INN/INP3, ADCx_INN/INP8, ADCx_INN/INP2, ADCx_INN/INP2, ADCx_INN/INP0, ADCx_INN/INP2, and ADCx_INN/INP2.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The

total number of conversions in the regular group must be written in the LEN[3:0] bits in the ADC_SQR1 register.

- An **injected group** is composed of up to four conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the JLEN[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while a regular conversion is ongoing. ADC regular conversions must consequently be stopped by setting ADSTP (refer to *Section 26.4.20: Stopping an ongoing conversion (ADSTP, JADSTP)*).

In addition, the ADC_JSQR register must not be modified while an injected conversion is ongoing. ADC injected conversions must consequently be stopped by setting JADSTP, except when JQDIS is cleared.

26.4.13 Channel preselection register (ADC_PCSEL)

The PCSEL bit of the ADC_PCSEL register controls the analog switch integrated in the I/O.

For each channel selected through SQRx or JSQRx bits, the corresponding PCSEL bit must be configured in advance in the ADC_PCSEL register. The ADC input multiplexer selects the ADC input according to SQRx and JSQRx configuration with very high speed. The analog switch integrated in the I/O cannot react as fast as the ADC multiplexer. To avoid the delay due to the analog switch control on the I/O, it is necessary to preselect the input channels that are selected through the SQRx and JSQRx. The selection is based on the V_{INP} of each ADC input. For example, if the ADC converts ADCx_INN/INP1, PCSEL1 bit must also be set in ADC_PCSEL register. If ADCx_INN/INP1 is configured as a differential channel, both PCSEL1 and PCSEL0 must be set, which corresponds to ADC1_INN1 and ADC2_INN1.

Note: Configuring the PCSEL bit is not necessary for the internal channels (such as V_{REFINT}).

26.4.14 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be sufficient for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time, which is programmable through the SMP[2:0] bits of the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 4.5 ADC clock cycles
- SMP = 010: 7.5 ADC clock cycles
- SMP = 011: 12.5 ADC clock cycles
- SMP = 100: 24.5 ADC clock cycles
- SMP = 101: 47.5 ADC clock cycles
- SMP = 110: 256.5 ADC clock cycles
- SMP = 111: 848.5 ADC clock cycles

The total conversion time is calculated as follows (12-bit mode):

$$T_{CONV} = \text{Sampling time} + 13 \text{ ADC clock cycles}$$

Example

With $F_{adc_ker_ck}$ = 32 MHz and a sampling time of 3 ADC clock cycles:

$$T_{CONV} = (3 + 13) \times \text{ADC clock cycles} = 16 \text{ ADC clock cycles} = 500 \text{ ns}$$

The ADC notifies the end of the sampling phase by asserting the EOSMP flag (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum and a maximum sampling times as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

The bulb sampling mode enables to obtain longer sampling time.

When the BULB bit is set in the ADC_CFGR2 register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPRx registers. The very first ADC conversion after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

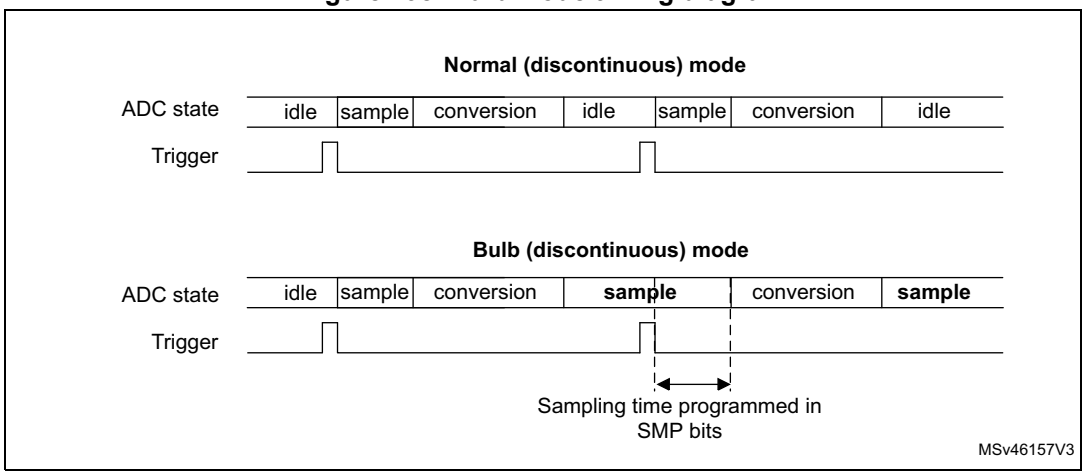
The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

Bulb mode is supported exclusively in regular conversion; dual mode is not supported.

The BULB bit can be modified only when the ADEN bit of the ADC_CR register is cleared.

When the BULB bit is set, it is not allowed to set the SMPTRIG bit in ADC_CFGR2.

Figure 198. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed through SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected (EXTEN[1:0] = 01), each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

Due to the synchronization mechanism, the minimum allowed pulse width is seven adc_ker_ck periods.

When a software trigger is selected (EXTEN[1:0] = 00), the software trigger is not the ADSTART bit of ADC_CR but the SWTRIG bit. The SWTRIG bit has to be set to start the sampling period, and it has to be cleared to end the sampling period and start the conversion. In this mode, the minimum sampling time is limited to seven ADC clock cycles.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is only compatible with the discontinuous mode. DISCNUM must be cleared for regular conversion mode. It is not supported for injected conversion mode and autoinjection mode.

When the SMPTRIG bit is set, setting the BULB bit is not allowed and only regular conversion is supported.

The sampling time control trigger mode is not compatible with the following dual modes: DUAL[4:0] = 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, and 0b01001.

26.4.15 Single conversion mode

To enable the single conversion mode for regular channels, clear both CONT and DISCEN bits in ADC_CFGR1 register. To enable it on injected channel, clear JDISCEN bit.

In single conversion mode, the ADC performs once all the regular conversions of the programmed channels. This mode is started by one of the following events:

- ADSTART bit set in the ADC_CR register (for a regular channel)
- JADSTART bit set in the ADC_CR register (for an injected channel)
- An external hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit 8-level FIFO accessible through the ADC_DR register,
- The EOC (end of regular conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 32-bit ADC_JDRy registers,
- The JEOC (end of injected conversion) flag is set, and
- An interrupt is generated if the JEOCIE bit is set.

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set, and
- An interrupt is generated if the JEOSIE bit is set.

The ADC then stops until a new external regular or injected trigger occurs or until the ADSTART or JADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

26.4.16 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts all the conversions of the sequence. This mode is started by setting the CONT bit of the ADC_CFGR1 register, either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the eight 32-bit word FIFO accessible through the ADC_DR register.
- The EOC (end of conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled, this means that it is forbidden to set both DISCEN and CONT bits.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit) (refer to Section : Autoinjection mode).

26.4.17 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN and CONT = 0 bit in the ADC_CFGR1 register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the LEN[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11

  - First trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - Second trigger: channels converted are 6, 7, 8 (an EOC event is generated at the end of each conversion).
  - Third trigger: channels converted are 9, 10, 11 (an EOC event is generated at the end of each conversion) and an EOS event is generated after the conversion of channel 11.
  - Fourth trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - First trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all devices.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the fourth trigger reconverts the channels 1, 2 and 3 in the first subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR1 register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels with DISCNUM = 0.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JLEN[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - First trigger: channel 1 converted (a JEOC event is generated)
  - Second trigger: channel 2 converted (a JEOC event is generated)
  - Third trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all devices.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the fourth trigger reconverts the first injected channel 1.

It is not possible to use both autoinjection mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

26.4.18 Starting conversions (ADSTART, JADSTART)

ADC regular conversions can be started by setting ADSTART.

When ADSTART is set, the conversion starts:

- Immediately if EXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected regular hardware trigger, if EXTEN[1:0] is not equal to 00.

The application software starts ADC injected conversions by setting JADSTART.

When JADSTART is set, the conversion starts:

- Immediately if JEXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected injected hardware trigger, if JEXTEN[1:0] is not equal to 00.

Note: In autoinjection mode (JAUTO = 1), use the ADSTART bit to start regular conversions followed by autoinjected conversions (JADSTART must be kept cleared).

When the hardware trigger is enabled, ADSTART and JADSTART also indicate whether an ADC operation is ongoing. The ADC can be reconfigured while ADSTART and JADSTART are both cleared, indicating that the ADC is idle.

In the case of software triggering, ADSTP and/or JADSTP must be set before ADC reconfiguration.

ADSTART is deasserted by hardware:

- In single mode with software regular trigger (CONT = 0, EXTEN = 0x0): at any end of regular conversion sequence (EOS assertion)
- In discontinuous mode (CONT = 0, DISCEN = 1, DISCNUM= x): at any of subgroup sequence
- In all cases (CONT and EXTEN don’t care), after execution of the ADSTP procedure asserted by the software

Note: In continuous mode (CONT = 1), ADSTART is not deasserted by hardware when EOS is asserted because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTEN ≠ 0x00), ADSTART is not deasserted by hardware when EOS is asserted to help the software that does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is deasserted by hardware:

- In single mode with software injected trigger (JEXTEN = 0x0): at the end of each injected conversion sequence (JEOS assertion)
- In discontinuous mode (JDISCEN = 1): at the end of any conversion
- In all cases (JEXTEN=x), after execution of the JADSTP procedure asserted by the software

Note: When the software trigger is selected, the ADSTART bit must not be set if the EOC flag is still high.

In bulb mode (BULB = 1), the previous end-of-conversion automatically starts the sampling, then setting ADSTART starts the conversion including the programmed sampling time.

In sampling time control trigger mode (SMPTRIG = 1), SWTRIG bit must be set to start the conversion.

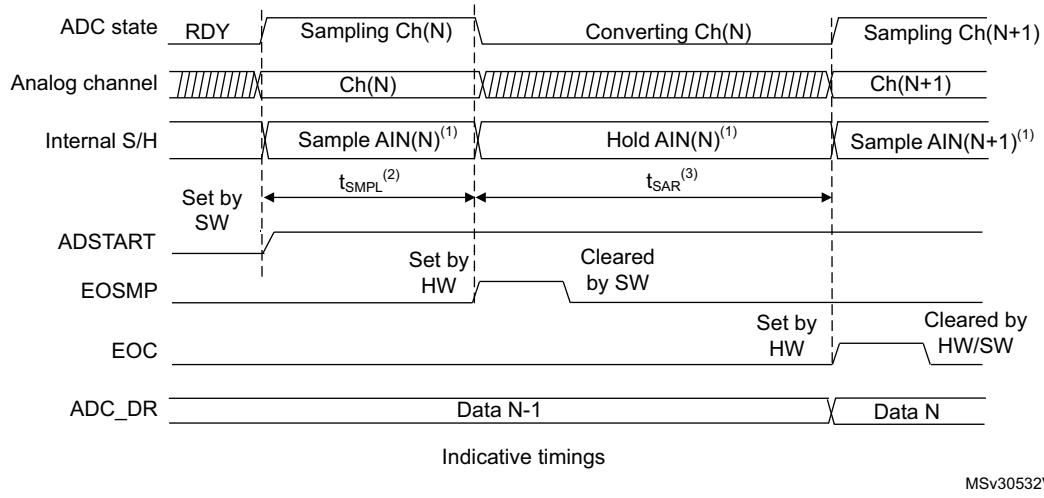
26.4.19 Timing

The elapsed time between the start of a conversion and the end of a conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$T_{CONV} = T_{SMPL} + T_{SAR} = [3\ _{|min} + 13\ _{|12bit}] \times T_{ADC_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 93.75\ ns\ _{|min} + 406.25\ ns\ _{|12bit} = 500\ ns$ (for $F_{adc_ker_ck} = 32\ MHz$)

Figure 199. Analog-to-digital conversion time



1. AIN represents the input voltage captured by the internal Sample and Hold (S/H) capacitor.
2. t_{SMPL} depends on SMP[2:0].
3. t_{SAR} depends on RES[2:0].

Note: The bulb and the sampling time control trigger modes are not described in Figure 199: Analog-to-digital conversion time.

26.4.20 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop ongoing regular conversions by setting ADSTP and ongoing injected conversions by setting JADSTP.

Stopping conversions resets the ongoing ADC operation. The ADC can then be reconfigured (for example by changing the channel selection or the trigger). It is then ready for a new operation.

Injected conversions can be stopped while regular conversions are still ongoing and vice versa. This allows, for instance, the reconfiguration of the injected conversion sequence and triggers while regular conversions are still ongoing (and vice versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with the partial result discarded (the ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (the ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC restarts a new sequence).

Once this procedure is complete, ADSTP/ADSTART bits (for regular conversion), or JADSTP/JADSTART bits (for injected conversion) are deasserted by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In autoinjection mode (JAUTO = 1), setting the ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 200. Stopping ongoing regular conversions

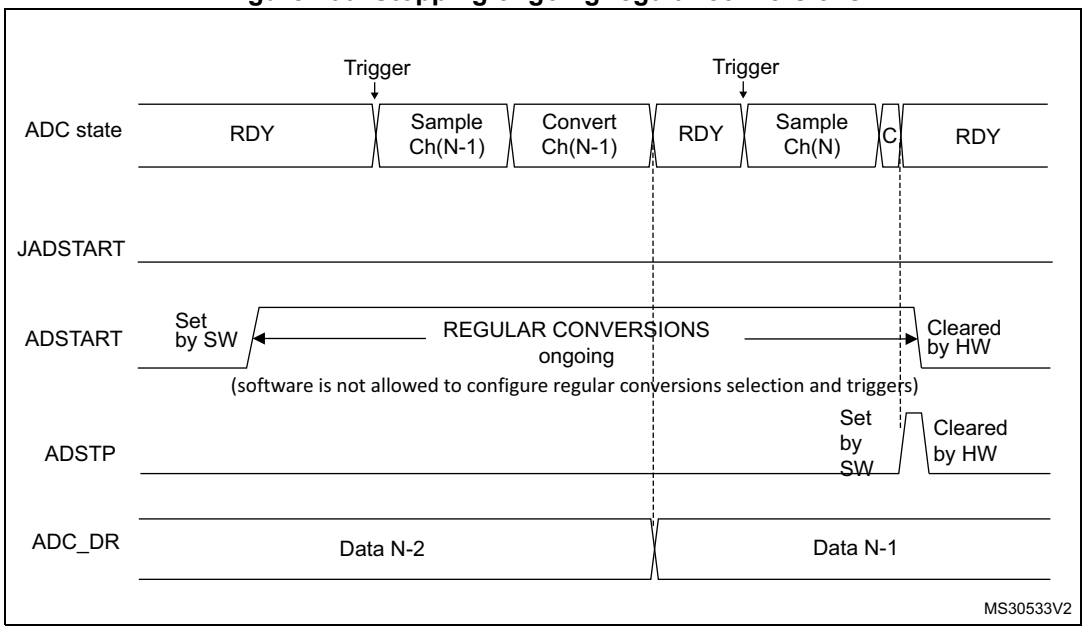
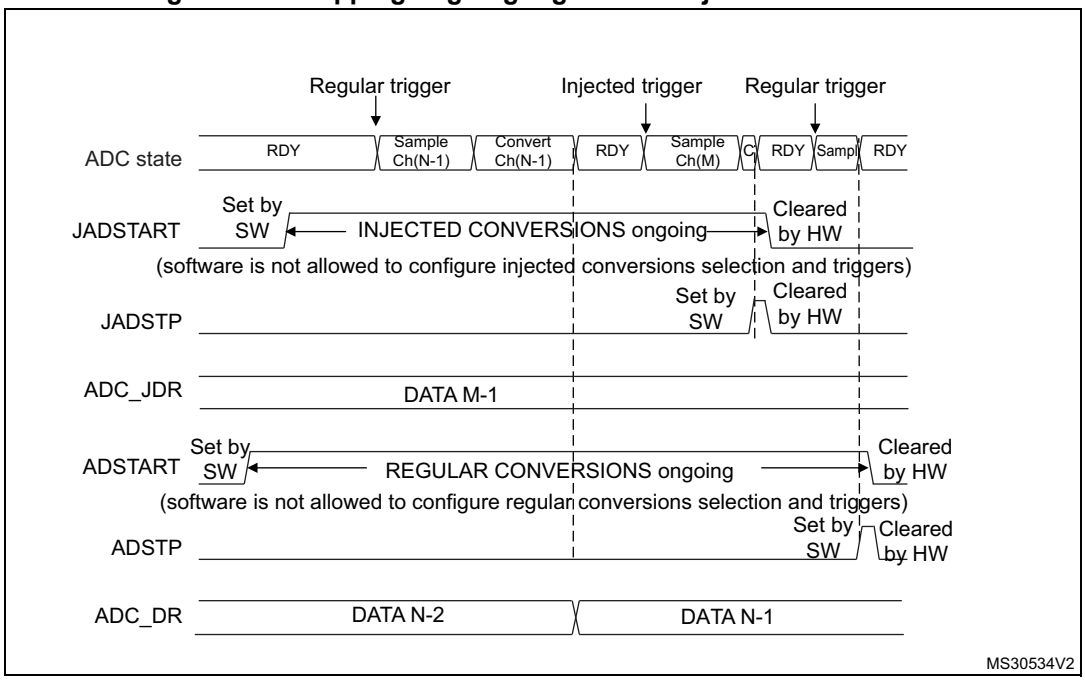


Figure 201. Stopping ongoing regular and injected conversions



26.4.21 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (for example timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

The regular trigger selection is effective once the software has set ADSTART while the injected trigger selection is effective once the software has set JADSTART bit.

All hardware triggers that occur while a conversion is ongoing are ignored:

- If ADSTART = 0, regular hardware triggers are ignored.
- If JADSTART = 0, injected hardware triggers are ignored.

Table 227 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 227. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware trigger detection disabled, software trigger detection enabled
01	Hardware trigger with detection on the rising edge (sampling time controlled in SMPTRIG mode)
10	Hardware trigger with detection on the falling edge
11	Hardware trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

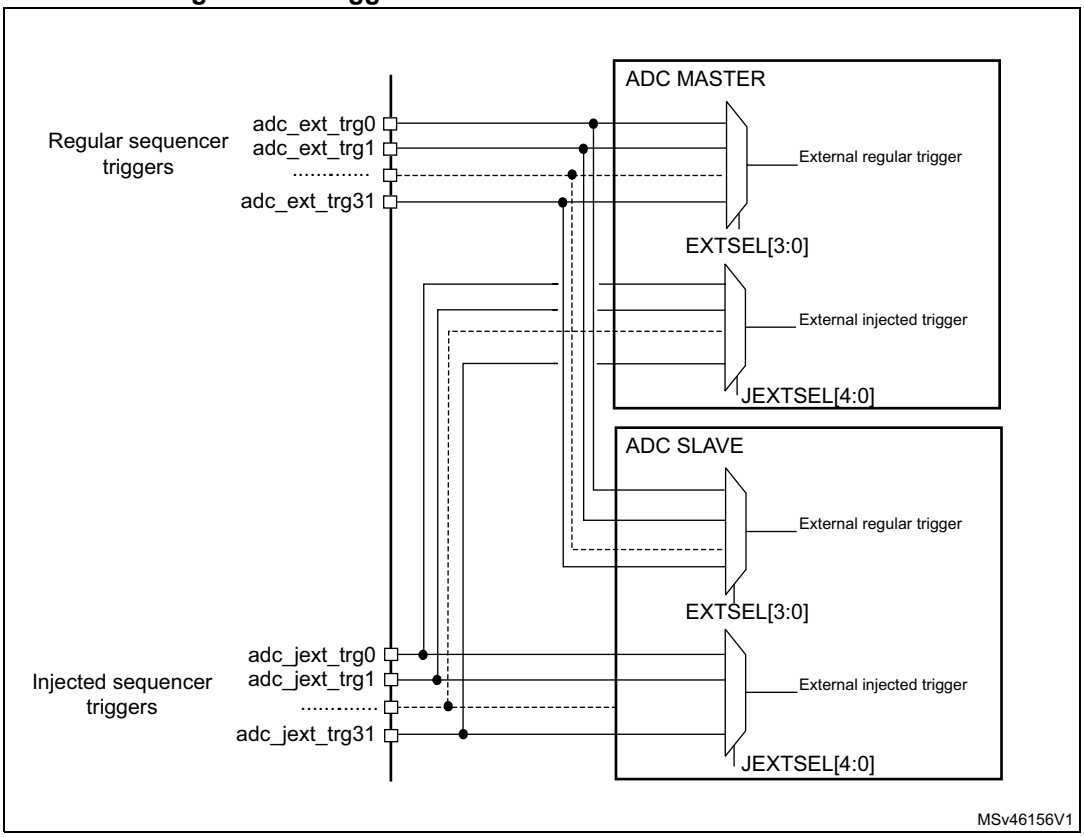
Table 228. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

The EXTSEL and JEXTSEL control bits select which, out of 32 possible events, can trigger the conversion of the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Figure 202. Triggers shared between ADC master and slave



Refer to *ADC interconnection* tables in *Section 26.4.2: ADC pins and internal signals* for the list of the external triggers.

26.4.22 Injected channel management

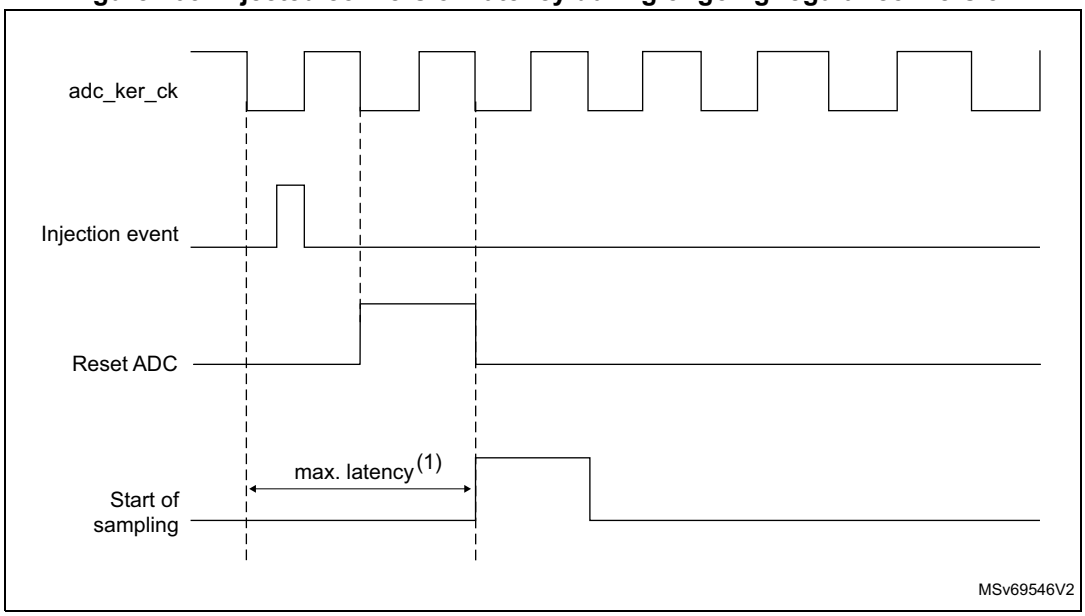
Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR1 register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once if JDISCEN = 0).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 203* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence.

Figure 203. Injected conversion latency during ongoing regular conversion



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

Autoinjection mode

If the JAUTO bit in the ADC_CFGR1 register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

Note: Autoinjection mode is compatible with discontinuous mode.

26.4.23 Queue of context for injected conversions

A queue of context is implemented to anticipate up to two contexts for the next injected sequence of conversions. JQDIS bit of the ADC_CFGR1 register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled. The queue of context is not supported in dual modes or in auto delay mode.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in the ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JLEN[1:0] in the ADC_JSQR register)

All the parameters of the context are defined in a single register ADC_JSQR and this register implements a queue of two buffers, allowing the bufferization of up to two sets of parameters:

- The ADC_JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written to the ADC_JSQR register is stored into the queue of context.
- At the beginning, the queue is empty and the first write access to the ADC_JSQR register defines the first context and the ADC is ready to receive injected triggers if JADSTART = 1.
- Once an injected sequence is complete, the queue is consumed and the context changes according to the next ADC_JSQR parameters stored in the queue. This new context is applied for the next injected sequence of conversions.
- A queue overflow occurs when writing to the ADC_JSQR register while the queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of ADC_JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two behaviors are possible when the queue becomes empty, depending on the value of the JQM control bit of the ADC_CFGR1 register:
  - If JQM = 0, the queue is empty just after enabling the ADC, but then it can never be empty during run operations: the queue always maintains the last active.
  - If JQM = 1, the queue becomes empty when end of injected sequence of last context occurs or if the queue is flushed by JADSTP = 1. When this occurs, there

is no more context in the queue and software needs to write new injected context in the ADC_JSQR register.

- Reading the ADC_JSQR register returns the current the ADC_JSQR context which is active at that moment. If JQM = 0, when the ADC_JSQR context is empty, the JSQi is read as 0x00.
- The queue is flushed when stopping injected conversions by setting JADSTP or when disabling the ADC by setting ADDIS:
  - If JQM = 0, the queue is flushed, but the last active context is maintained (ADC_JSQR register kept last value).
  - If JQM = 1, the queue becomes empty.

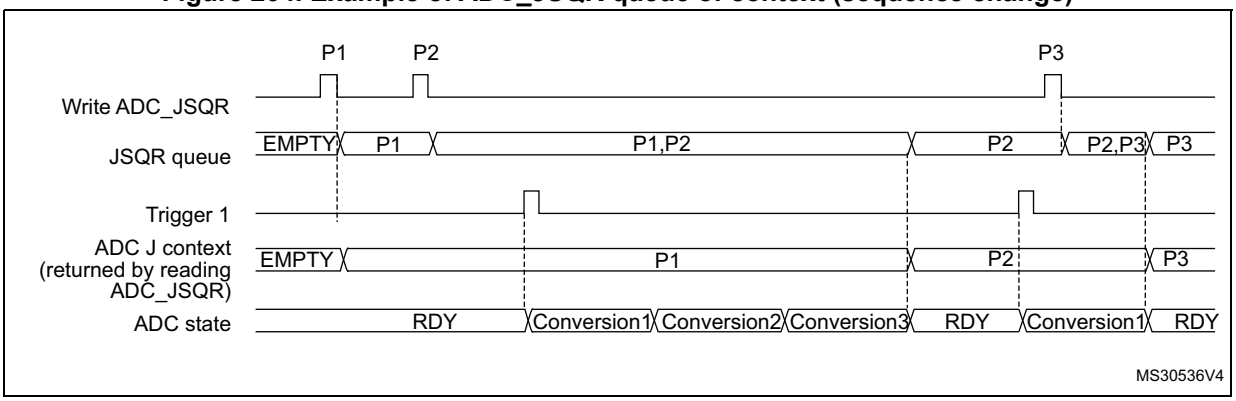
Note: When discontinuous mode (bit JDISCEN = 1) is configured, only JEOS, which marks the end of injected sequence, triggers the transfer of the queue to the active context (JSQR register). See example below (length = 3 for both contexts):

- *First trigger, discontinuous. Sequence 1: context 1, first conversion carried out.*
- *Second trigger, discontinuous. Sequence 1: context 1, second conversion carried out.*
- *Third trigger, discontinuous. Sequence 1: context 1, third conversion carried out. Then context change occurs to context 2.*
- *Fourth trigger, discontinuous. Sequence 2: context 2, first conversion carried out.*
- *Fifth trigger, discontinuous. Sequence 2: context 2, second conversion carried out.*
- *Sixth trigger, discontinuous. Sequence 2: context 2, third conversion carried out. Then context change occurs to next context.*

Behavior when changing the trigger or sequence context

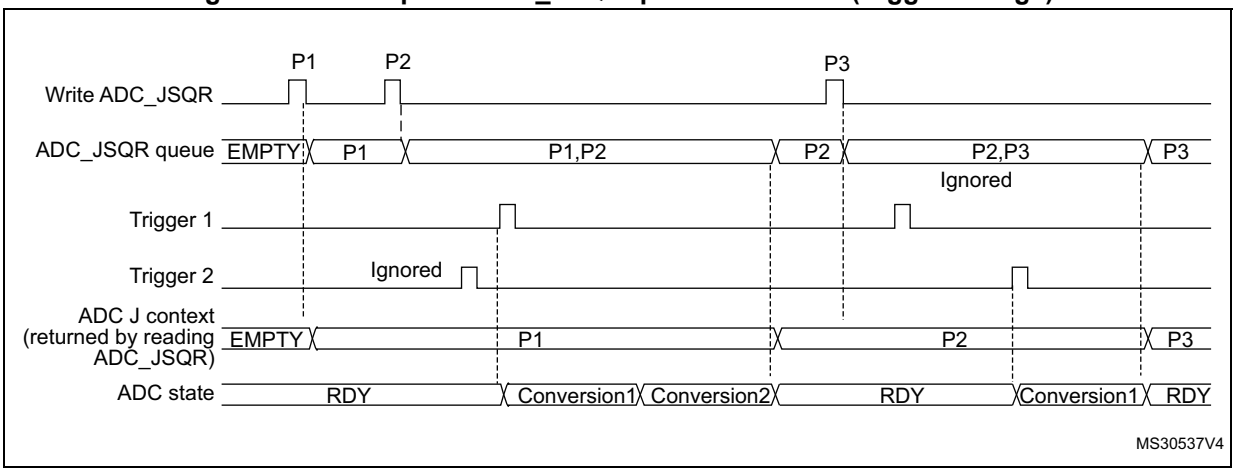
Figure 204 and *Figure 205* show the behavior of the context queue when changing the sequence or the triggers.

Figure 204. Example of ADC_JSQR queue of context (sequence change)



1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

Figure 205. Example of ADC_JSQR queue of context (trigger change)

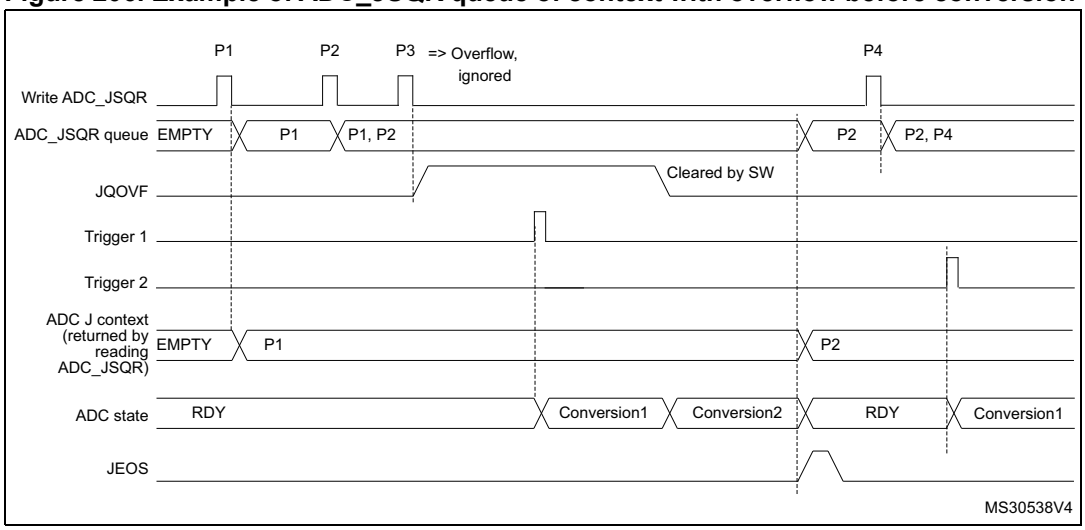


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

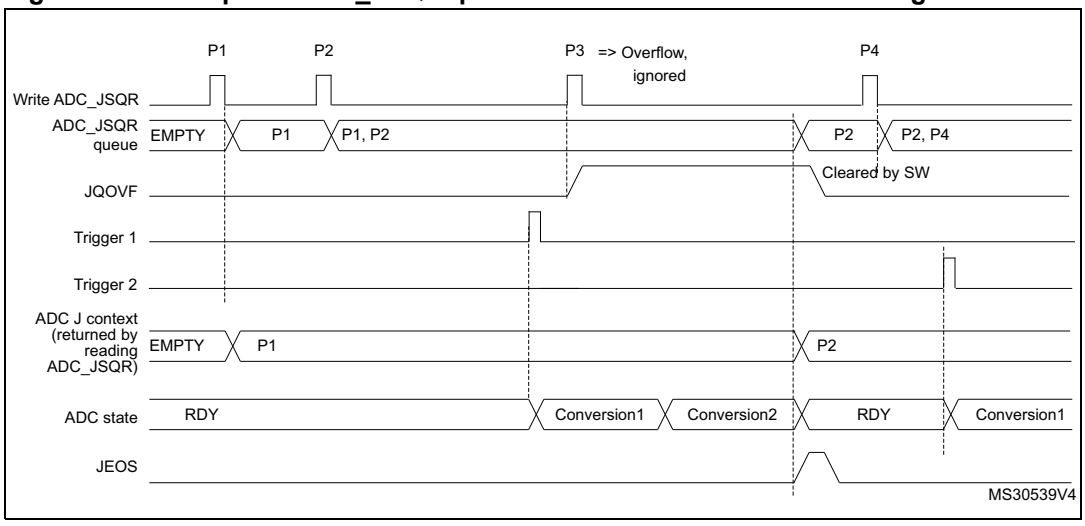
The *Figure 206* and *Figure 207* show the behavior of the context queue if an overflow occurs before or during a conversion.

Figure 206. Example of ADC_JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 207. Example of ADC_JSQR queue of context with overflow during conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

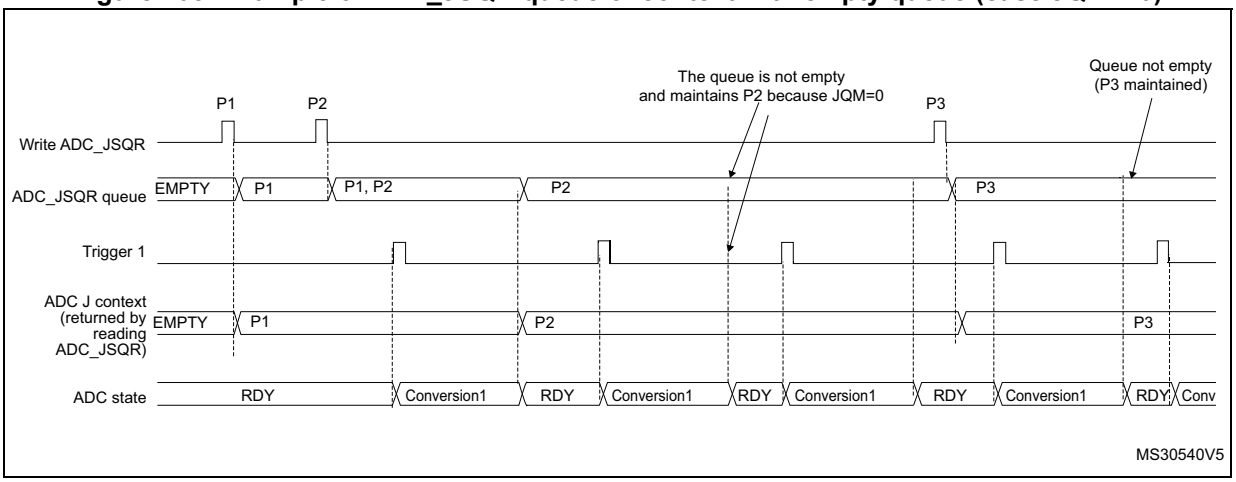
It is recommended to manage the queue overflows as described below:

- After each P context write to the ADC_JSQR register, the JQOVF flag shows if the write has been ignored or not (an interrupt can be generated).
- Avoid queue overflows by writing the third context (P3) only once the flag JEOS of P1 is generated, then the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 208 and *Figure 209* show the behavior of the context queue when the queue becomes empty in both cases JQM = 0 or 1.

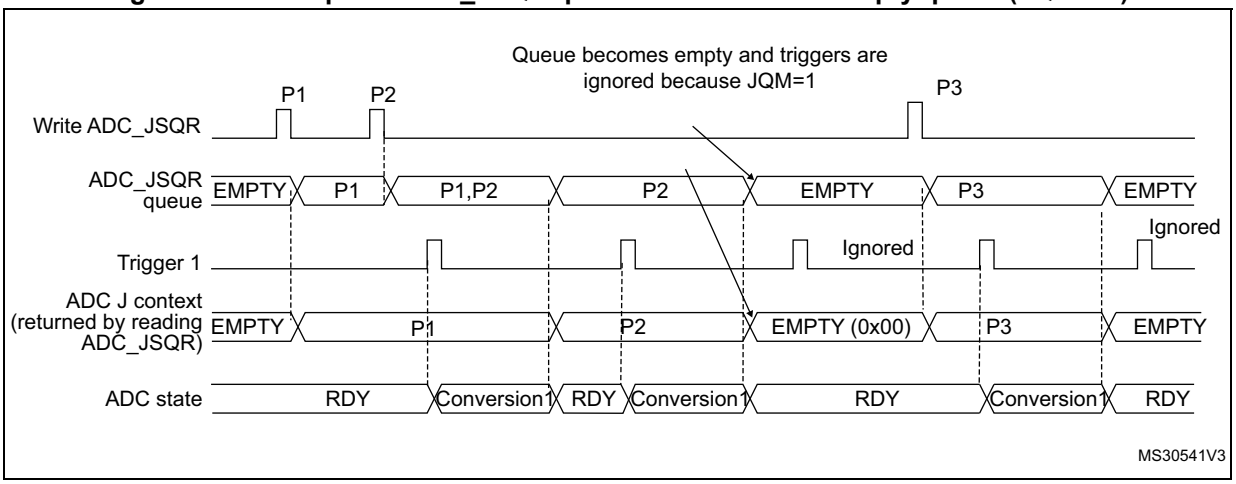
Figure 208. Example of ADC_JSQR queue of context with empty queue (case JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately as the context of P2 is already converted (JEOS is generated). However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately. It is recommended to program the new context before the JEOS timing, otherwise the previous context is used.

Figure 209. Example of ADC_JSQR queue of context with empty queue (JQM = 1)

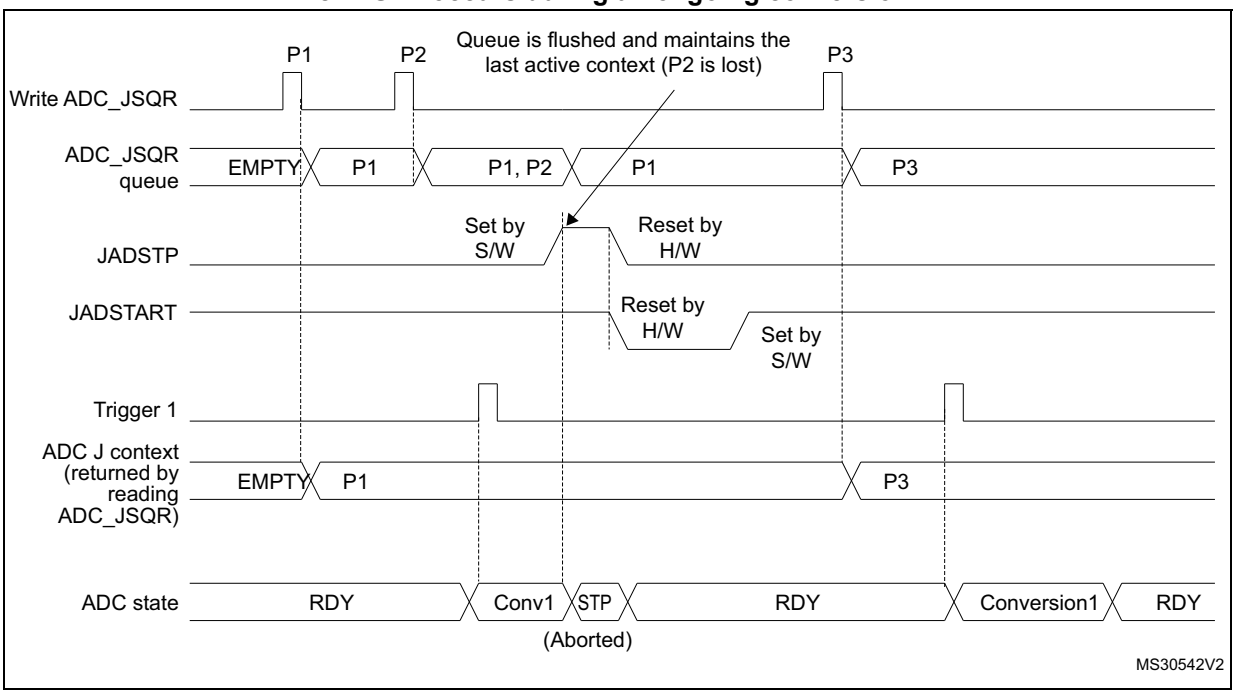


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

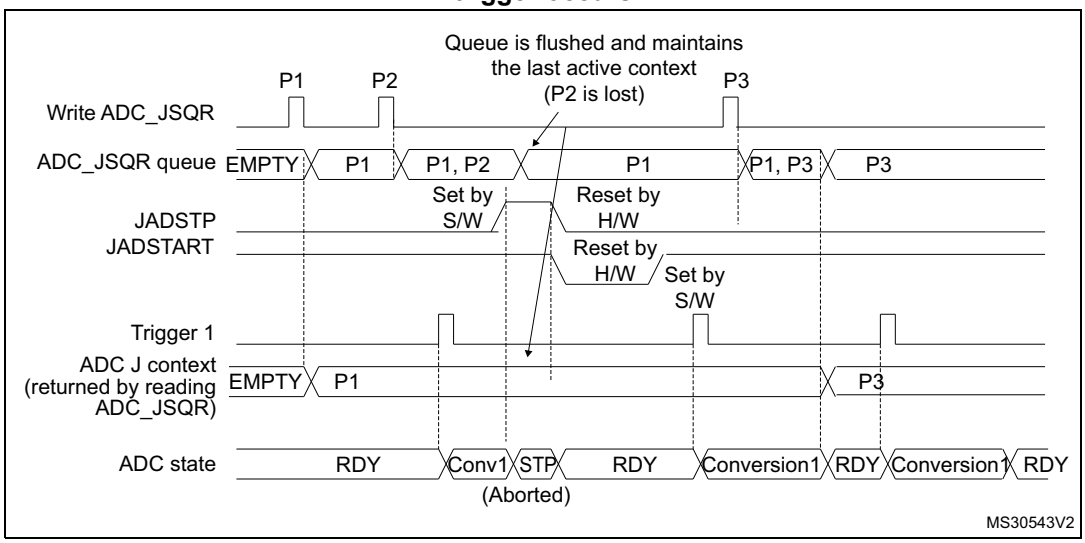
The figures below show the behavior of the context queue in various situations when the queue is flushed.

Figure 210. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion.



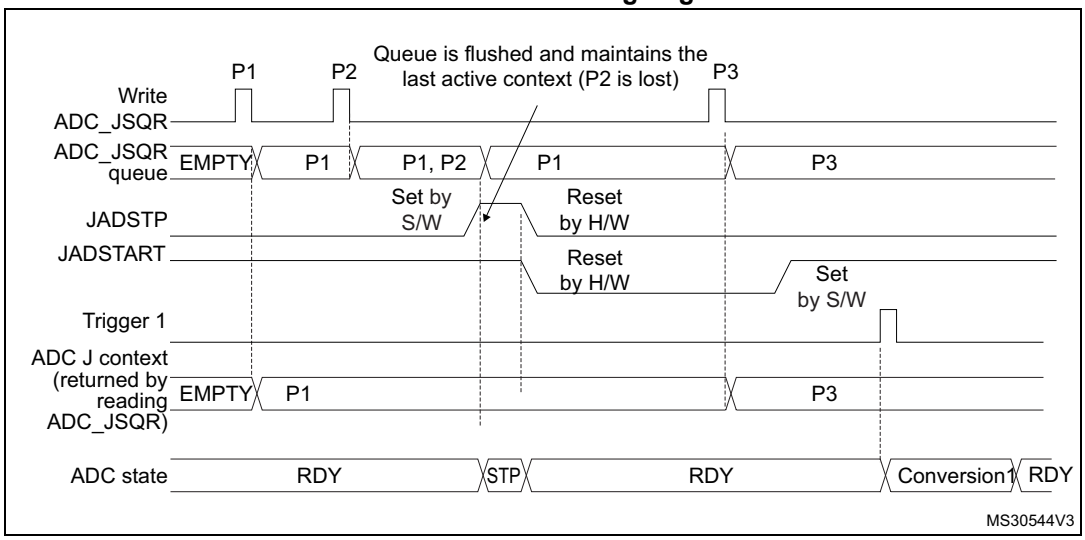
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 211. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs



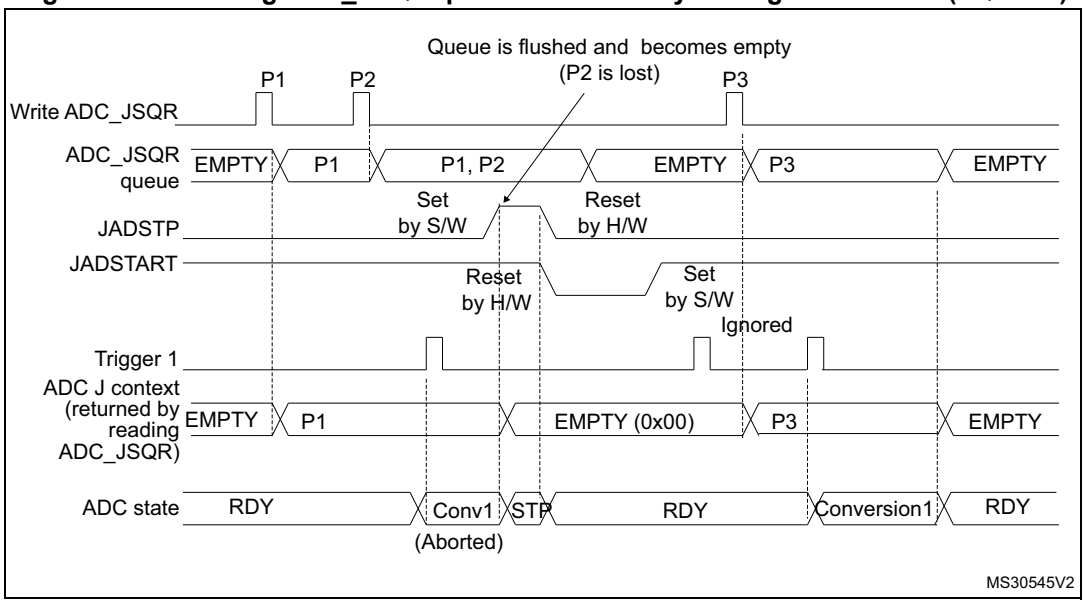
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 212. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion



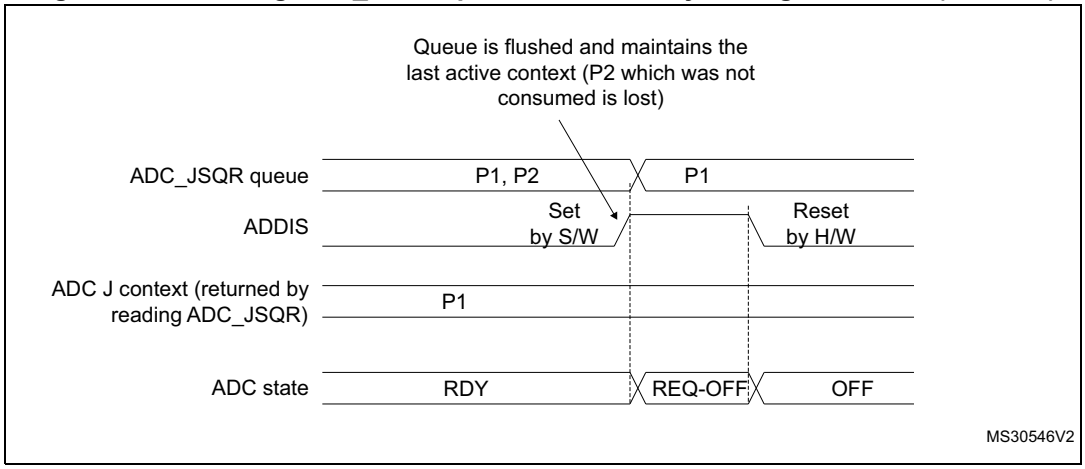
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 213. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 1)



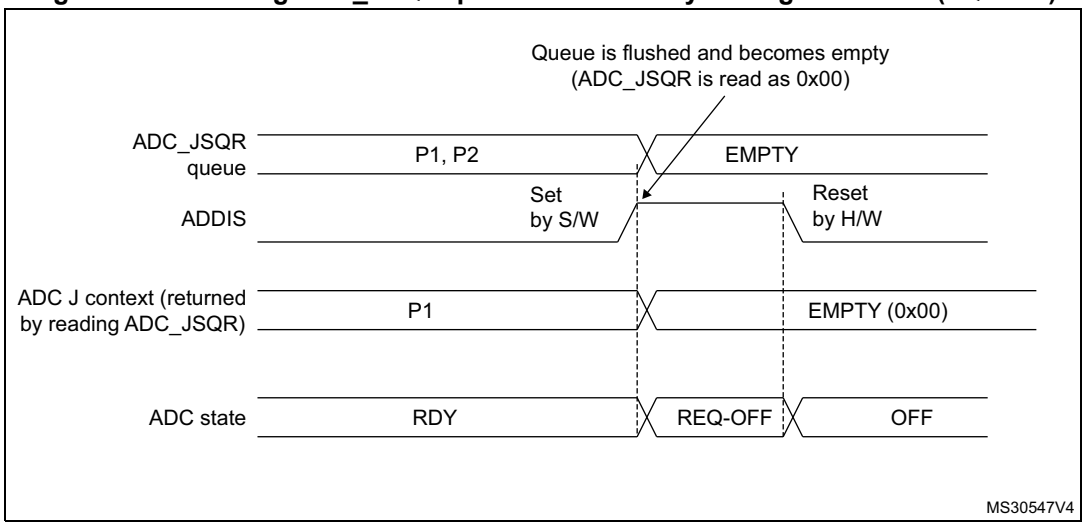
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 214. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 215. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 1)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Clear the context

- Setting JQDIS to 0 and then to 1 clears the context.
- Setting JADSTP to 0, writing a new context, and then setting JADSTART to 1 clears the previous context.

Disabling the queue

It is possible to disable the queue by setting the JQDIS bit of the ADC_CFGR1 register.

To clear the context queue, write 0 to JQDIS first, then write 1.

26.4.24 Programmable resolution (RES) - fast conversion mode

Faster conversions can be performed by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] control bits. *Figure 220*, *Figure 221*, *Figure 222* and *Figure 223* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 229*.

Table 229. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$ = 32 MHz	T_{CONV} (ADC clock cycles) (with Sampling time= 2.5 ADC clock cycles)	T_{CONV} (ns) at $F_{adc_ker_ck}$ = 32 MHz
12	13.5 ADC clock cycles	421.875 ns	16 ADC clock cycles	500.0 ns
10	11.5 ADC clock cycles	359.375 ns	14 ADC clock cycles	433.33 ns

Table 229. T_{SAR} timings depending on resolution (continued)

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$ = 32 MHz	T_{CONV} (ADC clock cycles) (with Sampling time= 2.5 ADC clock cycles)	T_{CONV} (ns) at $F_{adc_ker_ck}$ = 32 MHz
8	9.5 ADC clock cycles	296.875 ns	12 ADC clock cycles	366.67 ns
6	7.5 ADC clock cycles	156.25 ns	10 ADC clock cycles	300 ns

26.4.25 End of conversion and end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application at the end of each regular conversion (EOC) event and injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set. The software can clear the EOC flag either by programming it to 1 or by reading ADC_DR. If ADC_DR FIFO is not emptied, the EOC flag is set again and an interrupt can be generated again.

The ADC sets the JEOC flag as soon as new injected conversion data is available in one of the ADC_JDRy registers. An interrupt can be generated if the JEOCIE bit is set. The software can clear the JEOC flag either by programming it to 1 or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of the sampling phase by setting the EOSMP flag (for regular conversions only). The EOSMP flag is cleared by software by programming it to 1. An interrupt can be generated if the EOSMPIE bit is set.

26.4.26 End of conversion sequence (EOS, JEOS)

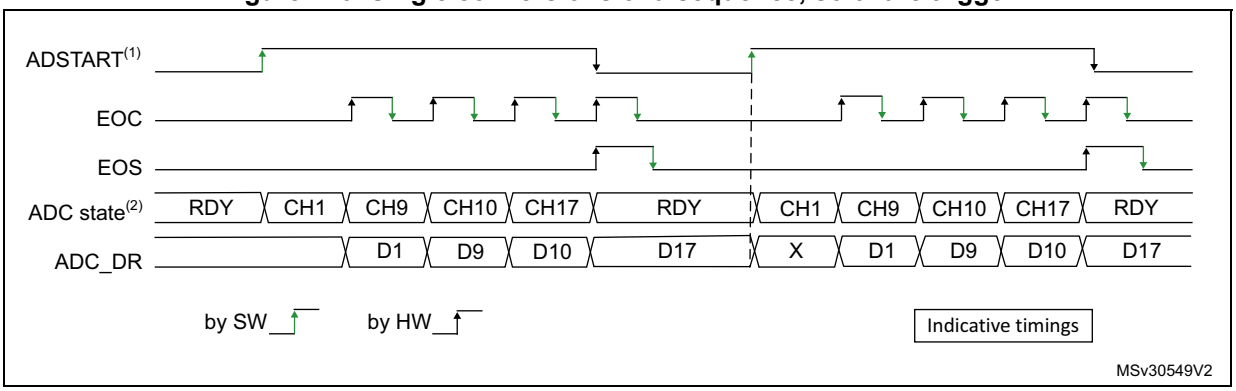
The ADC notifies the application at the end of each regular sequence (EOS) and injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the conversion of the last data of the regular sequence is complete. An interrupt can be generated if the EOSIE bit is set. The EOS flag is cleared by the software by programming it to 1.

The ADC sets the JEOS flag as soon as the conversion of the last data of the injected sequence is complete. An interrupt can be generated if the JEOSIE bit is set. The JEOS flag is cleared by the software by programming it to 1.

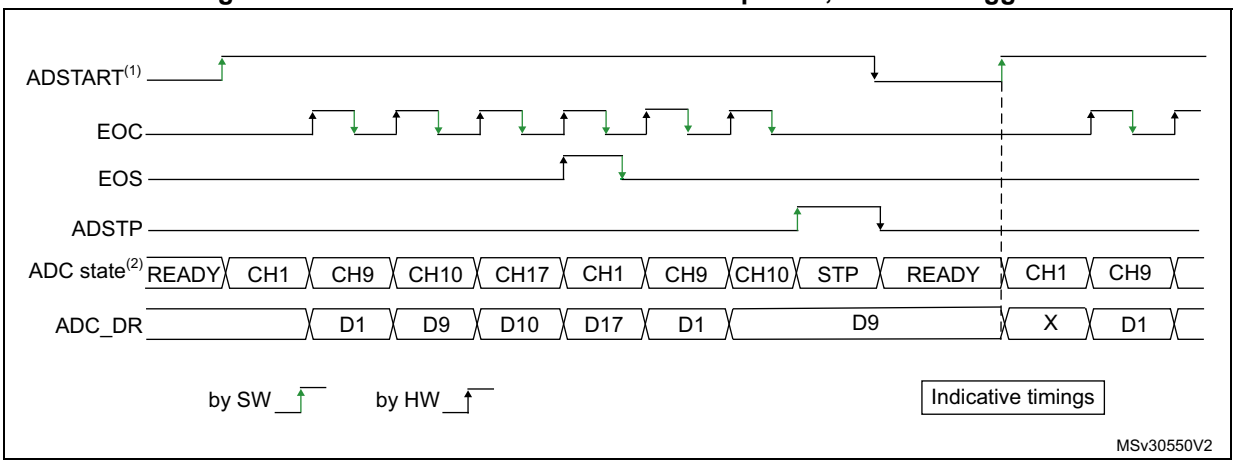
26.4.27 Timing diagram examples (single/continuous modes, hardware/software triggers)

Figure 216. Single conversions of a sequence, software trigger



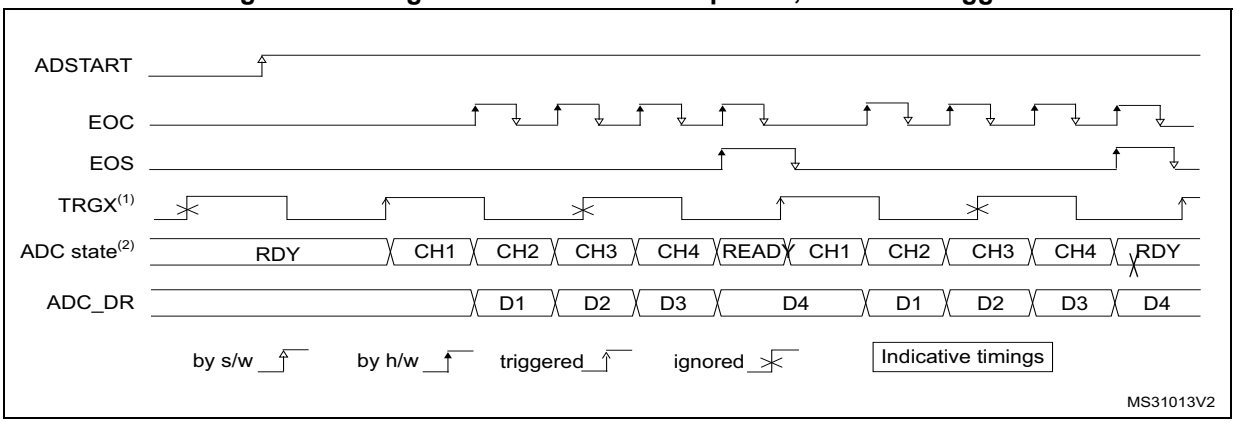
1. EXTEN[1:0] = 00, CONT = 0
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 217. Continuous conversion of a sequence, software trigger



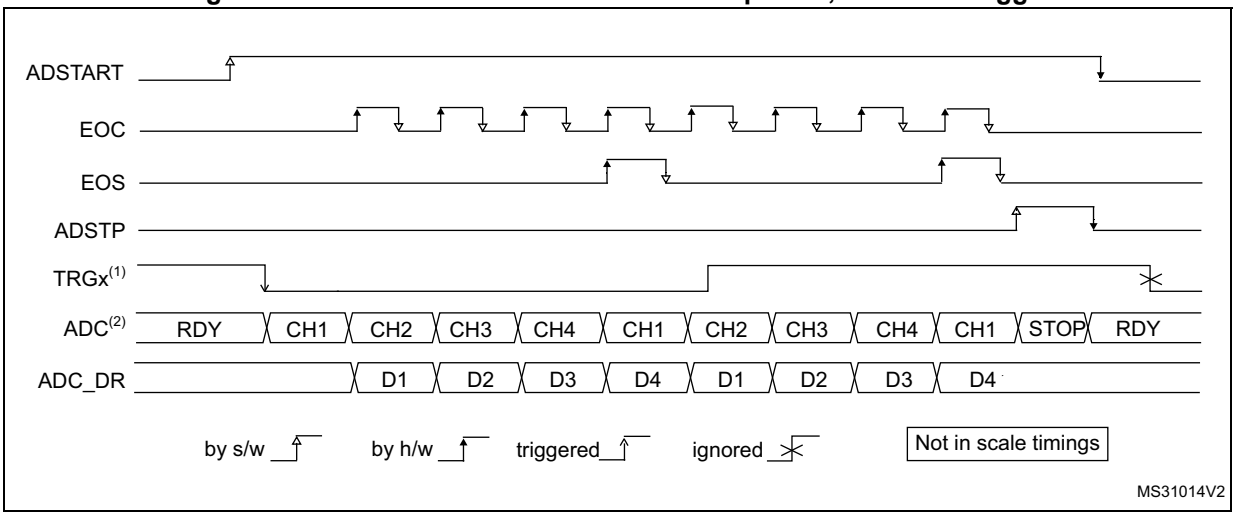
1. EXTEN[1:0] = 00, CONT = 1
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 218. Single conversions of a sequence, hardware trigger



1. TRGx (overfrequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 219. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1.
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

26.4.28 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS[a], JLSHIFT[a], USAT, SSAT, POSOFF)

Data and alignment

At the end of each regular channel conversion (when an EOC event occurs), the result of the converted data is stored in the 32-bit wide ADC_DR data register with FIFO (when OVRMOD = 0). Since the data register has eight levels of FIFO, the EOC flag is raised again as far as the FIFO is not emptied.

At the end of each injected channel conversion (when a JEOC event occurs), the result of the converted data is stored in the 32-bit wide corresponding ADC_JDRy data register.

The OVSS[3:0], LSHIFT[3:0], JOVSS[3:0], and JLSHIFT[3:0] bitfields in the ADC_CFGR2 register select the alignment of the data stored after the conversion. By default, data are right-aligned. Refer to *Figure 220*, *Figure 221*, *Figure 222*, and *Figure 223* for examples of data alignment.

Note: The data can be aligned in normal and oversampling mode. The OVSS[3:0] bitfield is only effective in oversampling mode.

JOVSS[3:0] and JLSHIFT[3:0] is reserved for injected conversions. Their usage is the same as OVSS[3:0] and LSHIFT[3:0], unless otherwise specifically described.

Offset

An unsigned offset value y (y = 1, 2, 3, 4) can be applied to a channel by programming a value different from 0 in the OFFSET[21:0] bitfield of the ADC_OFRy register. The channel to which the offset is applied is programmed to the OFFSETy_CH[4:0] bits of ADC_OFCFGRy register. The offset can be positive or negative depending on the value of POSOFF bit. When POSOFF is cleared, the converted value is subtracted by the user-defined offset written in OFFSET[21:0] bits. The result can be a negative value. The read data is consequently signed and the SEXT bit represents the extended sign value.

The offset value must be lower than the maximum conversion value (for example the maximum offset value is 0xFFF in 12-bit mode).

The offset can be used to convert unsigned data to signed data (for example the offset value is equal to 0x800 in 12-bit mode).

The offset correction is also supported in oversampling mode. In this mode, the offset is subtracted when all the oversampling operations are complete.

Note: If the OFFSET channel is selected but the offset value is 0x0000, this offset programming is ignored. If the same OFFSET channel is programmed in different registers multiple times, the first programmed register takes priority.

Table 235 describes how the computation is performed for all the possible ADC resolutions and offset settings.

a. Only for products that include this bit.

Table 230. Offset computation versus data resolution

Resolution (RES[1:0] bits)	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed or unsigned 24-bit data, right aligned to [11:0]	-
01: 10-bit	DATA[11:2],00	OFFSET[11:2]	Signed or unsigned 24-bit data, right aligned to [9:0]	The user must configure OFFSET[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	OFFSET[11:4]	Signed or unsigned 24 bit data right aligned to [7:0]	The user must configure OFFSET[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	OFFSET[11:6]	Signed or unsigned 24 bit data right aligned to [5:0]	The user must configure OFFSET[5:0] to 0b000000.

Figure 220, *Figure 221*, *Figure 222* and *Figure 223* show alignments for signed and unsigned data. JOVSS and JLSHIFT are not described here, but they behave the same as OVSS and LSHIFT.

Figure 220. Right alignment (offset disabled, unsigned value)

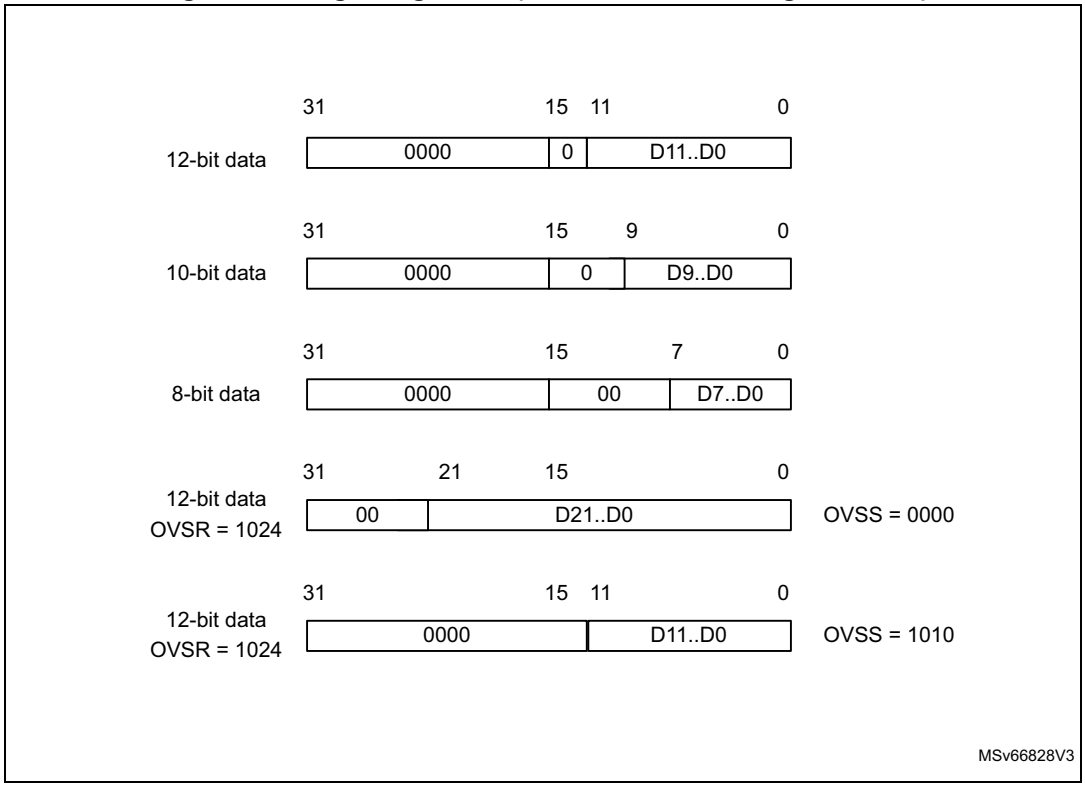


Figure 221. Right alignment (offset enabled, signed value)

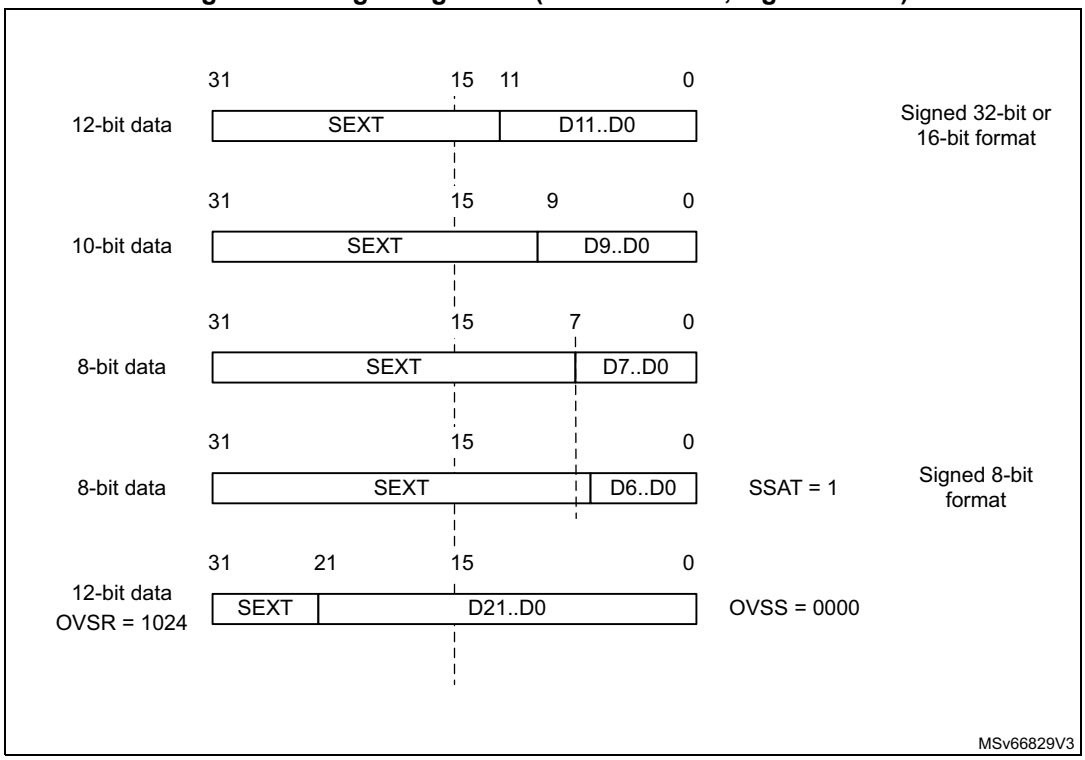


Figure 222. Left alignment (offset disabled, unsigned value)

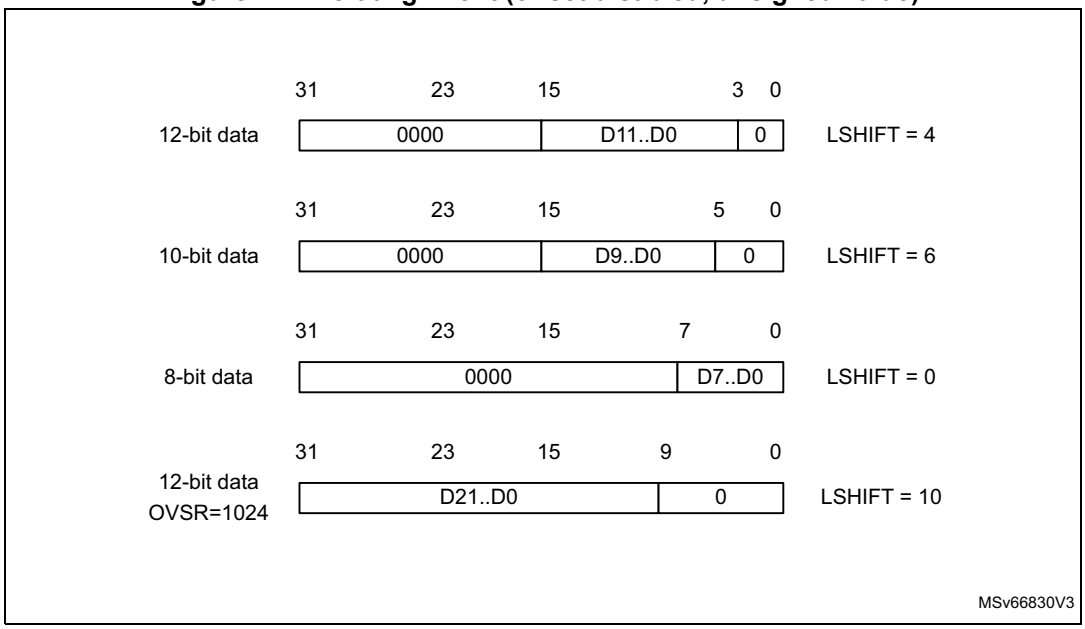
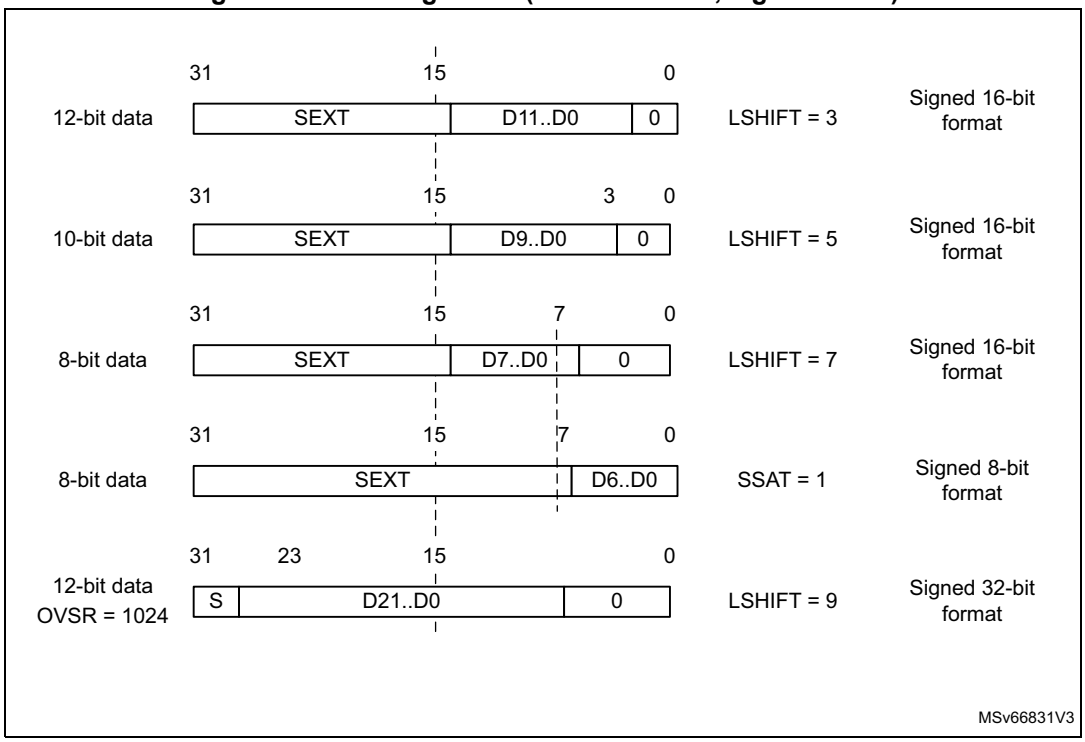


Figure 223. Left alignment (offset enabled, signed value)



Management of signed and unsigned saturation format (SSAT, USAT)

The offset correction might result in the data width to be wider than the original data.

To limit the original data width, the data saturation can be enabled through the SSAT and USAT bits of the ADC_OFCFGRy register.

Unsigned 12-bit data can be extended to 13-bit signed data by using an offset value different from 0x800.

The original data width can be preserved by setting SSAT bit to limit the data width to 12 bits.

Unsigned data can be saturated to the original data width by setting the USAT bit.

Table 231 shows the sign-extended data format corresponding to different resolutions.

Table 231.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	0	Sign-extended 13-bit significant data: SEXT[31:13] DATA[12:0]	0x0000 07FF - 0xFFFF F800
1	0	Sign-extended 12-bit significant data: SEXT[31:12] DATA[11:0]	0x7FF - 0x800

Table 231.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	1	Unsigned saturation 12-bit signification data: DATA[11:0]	0xFFF - 0x000
1	1	Reserved	-

Table 232 provides numerical examples for four different offset values.

Table 232. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0)

Raw conversion result	Offset value	Result SSAT = 0 USAT = 0	Result SSAT = 0 USAT = 1	Result SSAT = 1 USAT = 0
0xFFF	0x800	0x0000 07FF	0x07FF	0x07FF
0x800		0x0000 0000	0x0000	0x0000
0x000		0xFFFF F800	0x0000	0xF800
0xFFF	0x820	0x0000 07DF	0x07DF	0x07DF
0x800		0xFFFF FFE0	0x0000	0xFFE0
0x000		0xFFFF F7E0	0x0000	0xF7E0
0xFFF	0x7E0	0x0000 081F	0x081F	0x07FF
0x800		0x0000 0020	0x0020	0x0020
0x000		0xFFFF F820	0x0000	0xF820
0xFFF	0x020	0x0000 0FDF	0x0FDF	0x0FDF
0x800		0x0000 07E0	0x07E0	0x07E0
0x000		0xFFFF FFE0	0x0000	0xFFE0

Caution: SSAT must not be used in conjunction with USAT. No hardware check is performed to ensure that this condition is respected.

Gain compensation

When the GCOMP bit is set in the ADC_GCOMP register, the gain compensation is activated on all the converted data. After each conversion, data is calculated using the following formula.

$$DATA = DATA(\text{adc result}) \times (GCOMPCOEFF) / 4096$$

As GCOMPCOEFF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 3.999756.

Before storing the resulting data in RDATAx or JDATAx bitfields, the LSB - 1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for the oversampling. When the gain compensation is used in oversampling mode, the gain calculation is performed after the accumulation and

right-shift operations to minimize the power consumption (the gain calculation is done only once and not at each conversion).

The internal multiplier width is 32 bits and the input data width for the gain compensation must be less than 18 bits. When using oversampling with injected and regular conversion mode, the ROVSM bit of the ADC_CFGR2 register must be set to resume the pending conversion with the correct value.

When the gain compensation is activated, the ADC data latency is increased by one clock cycle. In continuous and scan conversion modes, the ADC conversion rate does not change.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) issues a notification that the regular converted data has not been read (by the CPU or the DMA) before the ADC_DR FIFO (eight stages) is overflowed, or adc_hclk clock is too slow to manage the data. If the CPU clears the EOC flag without reading the data register, the OVR flag may still be triggered.

The OVR flag is set when a new conversion completes while ADC_DR as single register or FIFO output is full. An interrupt is generated if the OVRIE bit is set.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP. The OVR flag is cleared by software by programming it to 1.

Data can be configured to be preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0

  The overrun event preserves the data register from being overwritten: the old data are maintained up to ADC_DR FIFO depth (eight data) and the new conversion is discarded and lost. If OVR remains at 1, further conversions occur but the result data is also discarded. The FIFO can be emptied through ADC_DR read access by the CPU while the OVR flag is set.
- OVRMOD = 1

  The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversion is performed normally and the ADC_DR register always contains the latest converted data. In this mode, DMA burst mode cannot be supported.

Figure 224. Example of overrun (OVRMOD = 0)

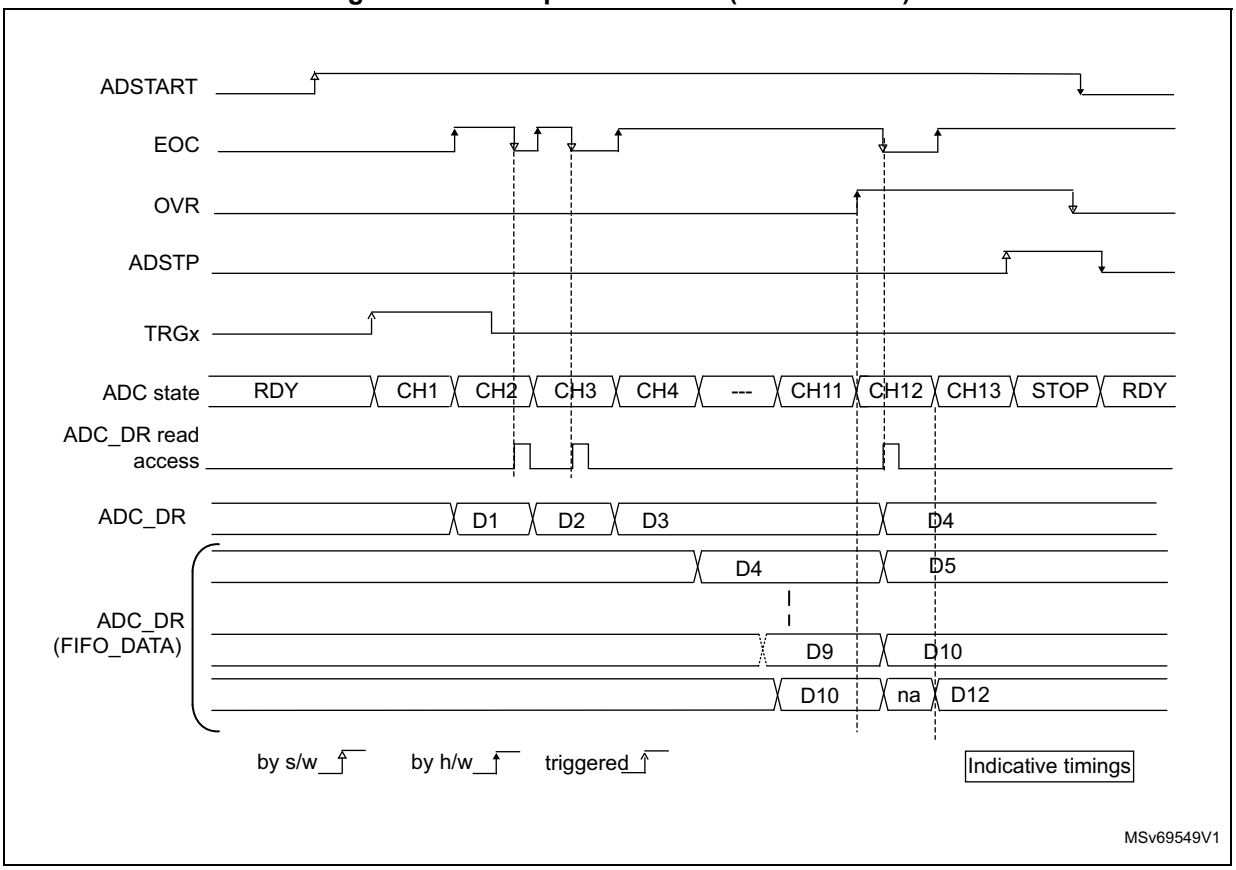
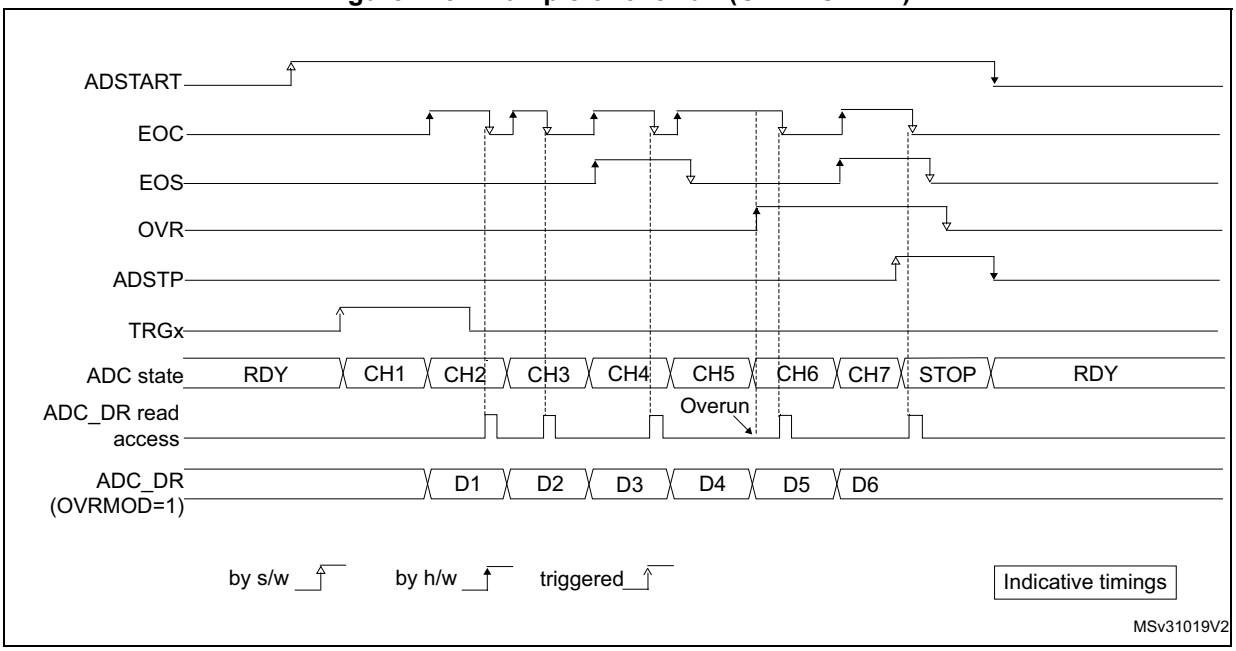


Figure 225. Example of overrun (OVRMOD = 1)



Note: Even if overrun may occur for injected channels, there is no overrun detection on these channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as errors.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and the OVR flag must be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

When the DMA mode is enabled (DMNGT[1:0] = 01 or 11 in the ADC_CFGR1 register in single ADC mode), a DMA request is generated after each channel conversion. This allows the transfer of the converted data from the ADC_DR register to the destination location configured in the DMA.

Despite this, if an overrun occurs (OVR = 1) because the DMA cannot serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA, which means that all the data transferred to the RAM can be considered as valid. (OVRMOD = 0).

When OVRMOD = 1, even if DMA does not transfer the data immediately, new data overwrite ADC_DR, causing DMA to potentially skip data. As a result, some conversion results may be missed in the data transferred to RAM.

Depending on the configuration of the OVRMOD bit, the data are either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*). If OVRMOD = 0 (data preserved), DMA transfer requests are blocked.

DMA can be configured through the DMNGT[1:0] bitfield of the ADC_CFGR1 register. Refer to *Section 26.4.33: Dual ADC modes* for information on DMA usage with dual ADC mode.

- DMA one-shot mode (DMNGT[1:0] = 01)

  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT[1:0] = 11)

  This mode is suitable when programming the DMA when the number of transfers is unknown.

DMA one-shot mode (DMNGT[1:0] = 01)

In one-shot mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (that is when DMA_LACK interrupt occurs, see *Direct-memory controller* section) even if a new conversion has been started.

When DMA transfers are complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- All ongoing conversions are aborted with the partial result discarded.
- The scan sequence is stopped and reset.
- The DMA and ADC are stopped.

DMA circular mode (DMNGT[1:0] = 11)

In circular mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register. This allows the configuration of the DMA in circular mode to handle a continuous analog input data stream.

DMA single-transfer or burst mode with FIFO (OVRMOD = 0)

The output data register features an eight-stage FIFO. Two different DMA requests are generated in parallel. When a data is available, an “SREQ single request” is generated. When four data are available, a “BREQ burst request” is generated. The DMA can be programmed either in single-transfer mode or in burst mode (four beats). The appropriate request line is selected by the DMA according to the mode selected. Refer to *Section Direct memory access controller* for further information.

Due to the existence of FIFO, DMA requests are generated until the FIFO is emptied. To stop DMA requests even when the FIFO is not emptied, set the ADSTP bit.

DMA for dual mode

Refer to *Section 26.4.33: Dual ADC modes*.

26.4.29 Dynamic low-power features

Autodelayed conversion mode (AUTDLY)

The ADC implements an autodelayed conversion mode controlled by the AUTDLY configuration bit in ADC_CFGR1. Autodelayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there is a risk of encountering an ADC overrun.

Regular conversion mode

When AUTDLY is set, a new conversion can start if the following conditions are met:

- The ADC_DR register is read or the EOC bit is cleared (see *Figure 226*).

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data and prevent overrun errors.

The delay is inserted after each regular conversion.

Injected conversion mode

When AUTDLY is set, a new conversion can start only if the JEOS bit has been cleared (see *Figure 227*). Since there are four ADC_JDRy registers, the autodelay is performed at the end on the injected conversion sequence.

There is no delay inserted between each conversion of the injected sequence, except after the last one.

A hardware trigger event occurring during this delay is ignored.

No delay is inserted between regular and injected conversions. The autodelay for regular and injected modes is managed separately.

Caution: The behavior is slightly different in autoinjection mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 230*).

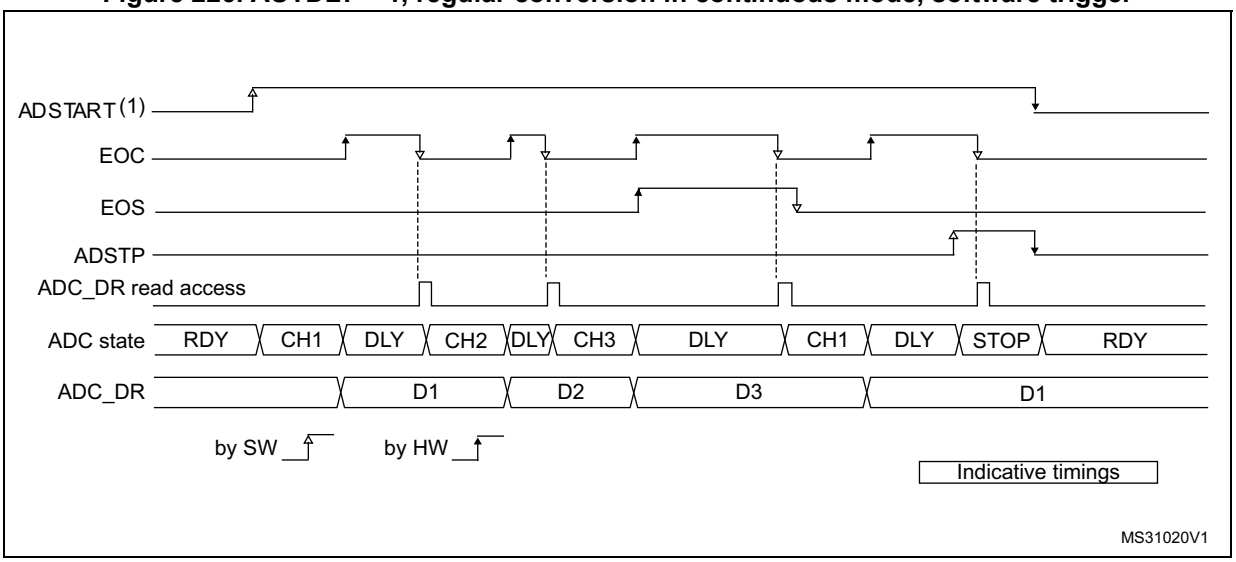
In autoinjection mode, there is no delay between the last regular conversion and the first injected conversion. The user must clear the last EOC flag before clearing the JEOS flag, otherwise the next sequence might start before EOC flag is cleared.

In autodelay mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In autodelay mode with JAUTO = 0, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

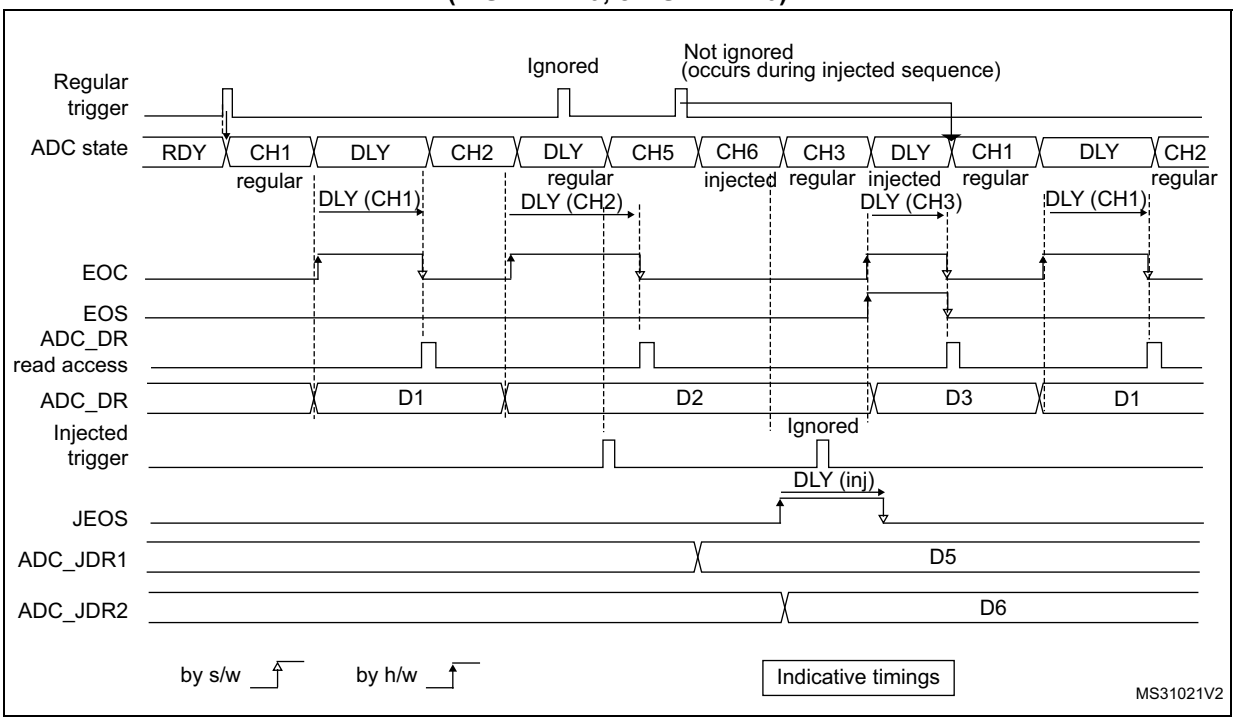
Note: AUTDLY mode is not supported with injected queue, SMPTRIG, or BULB mode.

Figure 226. AUTDLY = 1, regular conversion in continuous mode, software trigger



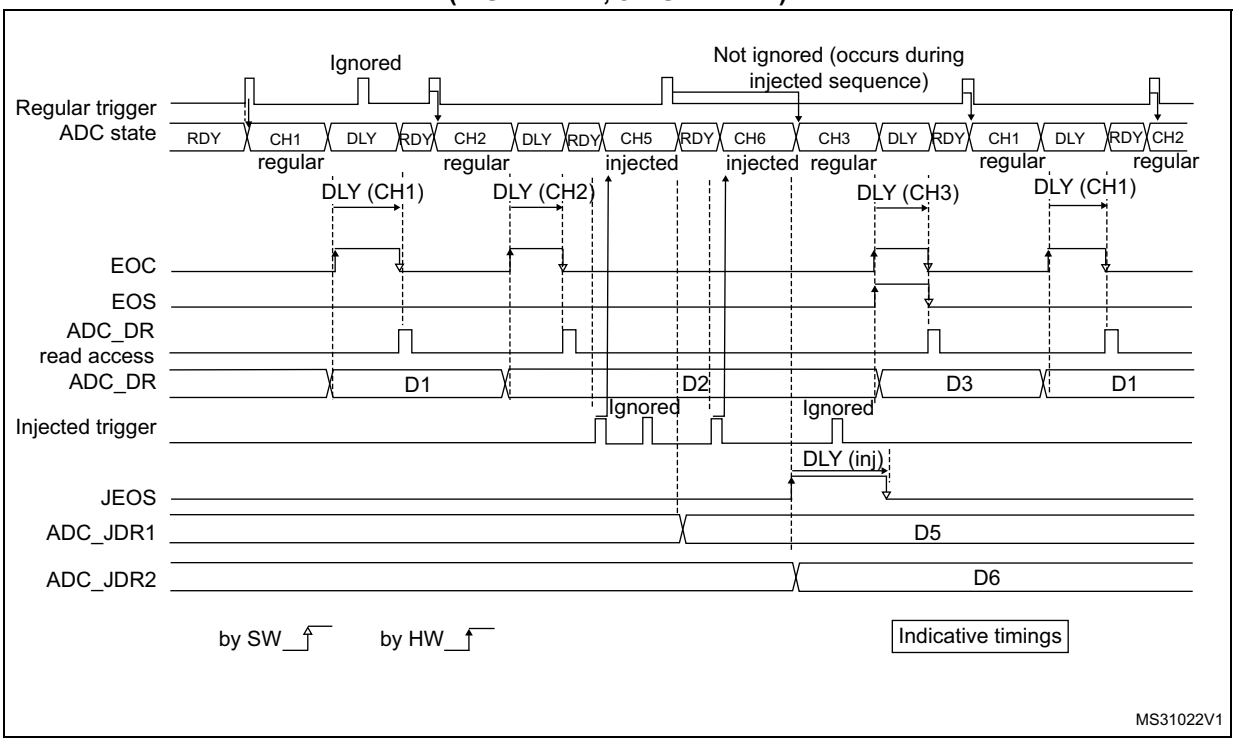
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, CHANNELS = 1, 2, 3.
3. Injected configuration DISABLED.

Figure 227. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



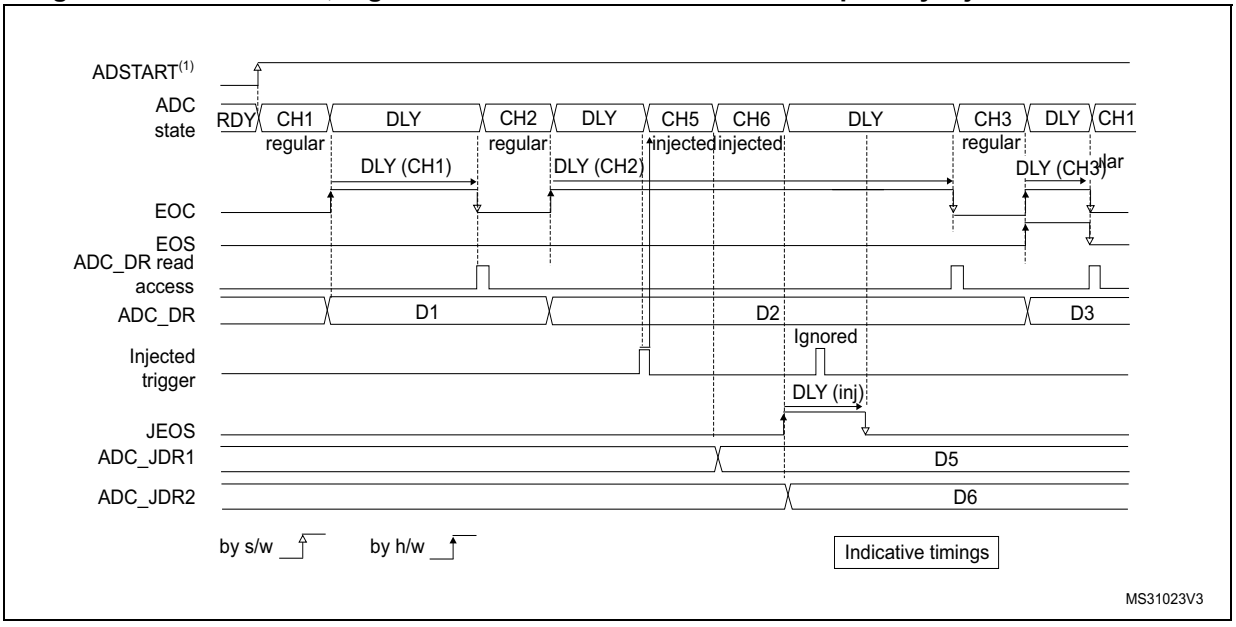
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 228. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



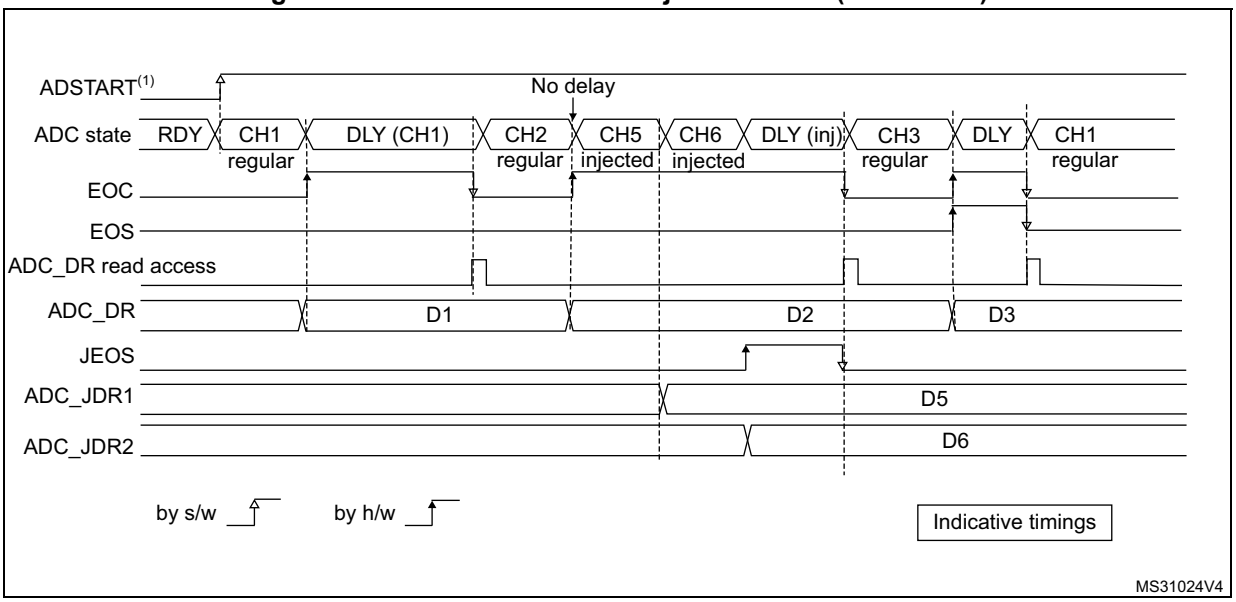
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 1, CHANNELS = 5, 6.

Figure 229. AUTDLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 230. AUTDLY = 1 in autoinjection mode (JAUTO = 1)



1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2.
3. Injected configuration: JAUTO = 1, CHANNELS = 5, 6.

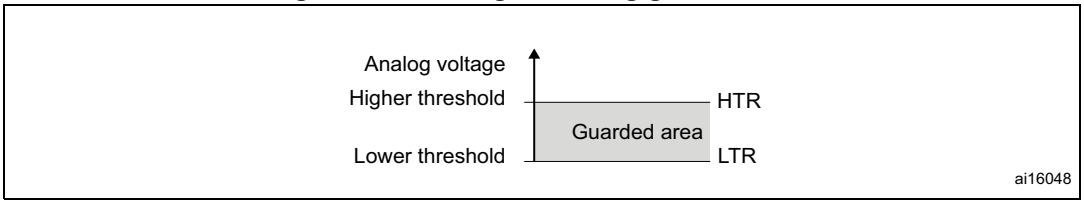
26.4.30 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)

Table 233. Analog window watchdog features

Feature	AWD1	AWD2	AWD3
Enable control	AWD1EN or JAWD1EN	ADC_AWD2CR	ADC_AWD3CR
Channel selection	AWD1CH[4:0]/AWD1SGL	ADC_AWD2CR	ADC_AWD3CR
Channel selection limitation	One or all channels	Any number of channels	
	Regular or injected	-	-
Threshold control	ADC_AWD1LTR/ ADC_AWD1HTR	ADC_AWD2LTR/ ADC_AWD2HTR	ADC_AWD3LTR/ ADC_AWD3HTR
Filter configuration	AWDFILT[2:0] in ADC_AWDyHTR		
Interrupt	AWDy in ADC_ISR		

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 231. Analog watchdog guarded area



AWDy flag and interrupt

An interrupt can be enabled for each of the three analog watchdogs by setting AWDyIE in the ADC_IER register (y = 1, 2, 3).

AWDy (y = 1, 2, 3) flag is cleared by software by writing 1 to it, or by setting ADDIS.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

AWD source channel register

The software can identify which ADC channel has generated the AWDy flag by reading the AWDS_CH[19:0] bitfield of the ADC_AWDySR register: AWDS_CH*i* bit is set to indicate that channel *i* has generated the flag. It is cleared by writing 1 to it or by disabling the ADC through setting the ADDIS bit.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN or JAWD1EN bit in the ADC_CFGR1 register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

Table 234 shows how the ADC_CFGR1 registers must be configured to enable the analog watchdog on one or more channels.

Table 234. Analog watchdog 1 channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog flag is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed through HTR[22:0] bits of the ADC_AWD1HTR register and LTR[22:0] bits of the ADC_AWD1LTR register for the analog watchdog 1.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024, offset and gain compensation)

When converting data with a resolution of less than 12 bits (defined by RES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data.

Table 235 describes how the comparison is performed for all the possible resolutions and for analog watchdog 1.

Table 235. Analog watchdog 1, 2, 3 comparison

Resolution (RES[1:0] bits)	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTR[22:0] and HTR[22:0]	-
01: 10-bit	DATA[11:2],00	LTR[22:0] and HTR[22:0]	The user must configure LTR[1:0] to 0b00 and HTR[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	LTR[22:0] and HTR[22:0]	The user must configure LTR[3:0] to 0b0000 and HTR[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	LTR[22:0] and HTR[22:0]	The user must configure LTR[5:0] to 0b000000 and HTR[5:0] to 0b000000.

1. Refer to *Section : Gain compensation* for additional details on analog watchdog comparison.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCH[19:0] (of ADC_AWD2CR and ADC_AWD3CR).

The corresponding watchdog is enabled when any bit of AWDCH[19:0] (of ADC_AWD2CR and ADC_AWD3CR) is set.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024 and offset compensation in signed format with gain compensation) and is programmed through the ADC_AWD2HTR, ADC_AWD2LTR, ADC_AWD3HTR and ADC_AWD3LTR registers.

When converting data with a resolution of less than 12 bits (defined by RES[1:0]) bits), the LSBs of the programmed threshold must be kept cleared, the internal comparison being performed on the full 12-bits converted data (left aligned).

ADCx_AWDy_OUT signal output generation

Each analog watchdog is associated with an internal hardware signal ADC_AWDy_OUT (*y* being the watchdog number), which is directly connected to the ETR input (external trigger) of some of the on-chip timers. Refer to the on-chip timers sections to understand how to select the ADC_AWDy_OUT signal as ETR.

ADC_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDy_OUT is reset after the end of the next guarded conversion that is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADC_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS). Note that stopping regular or injected conversions (by setting ADSTP or JADSTP) has no influence on the generation of ADCx_AWDy_OUT.

Note: The AWDy flag is set by hardware and reset by software: the AWDy flag has no influence on the generation of ADC_AWDy_OUT (ex: ADC_AWDy_OUT can toggle while the AWDy flag remains at 1 if the software did not clear the flag).

The AWDy flag is cleared by programming it to 1 or by set ADDIS bit. ADSTP and JADSTP bits do not reset the AWDy flag.

Figure 232. ADC_AWDy_OUT signal generation (on all regular channels)

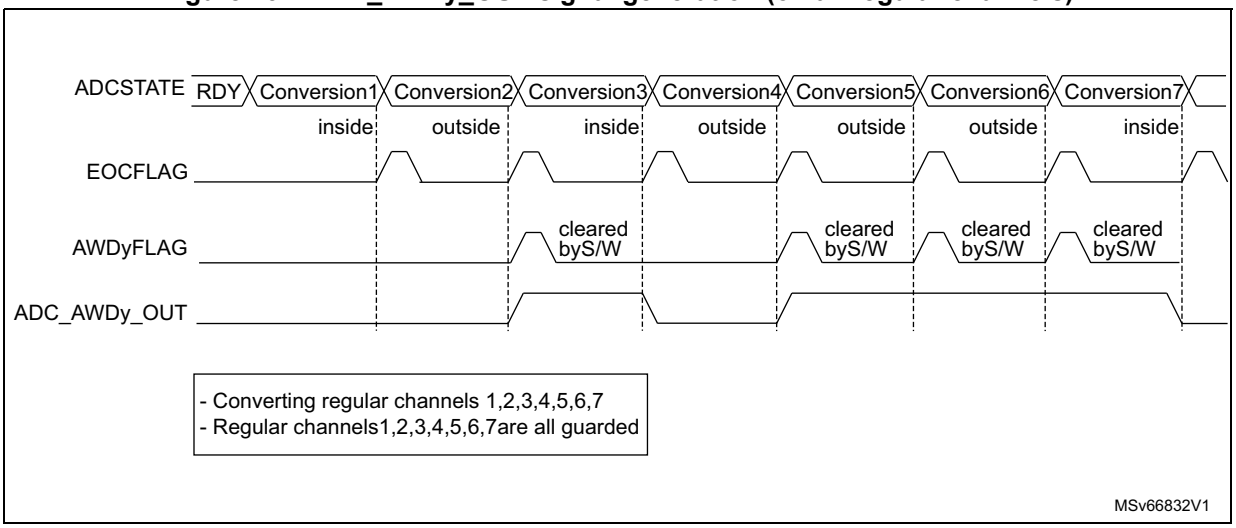


Figure 233. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software)

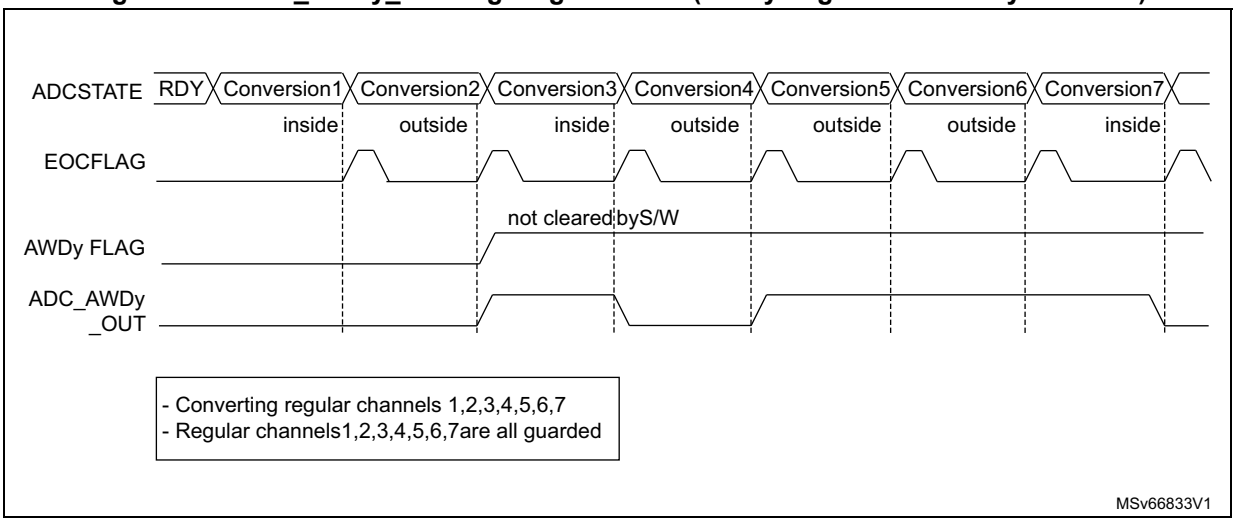


Figure 234. ADC_AWDy_OUT signal generation (on a single regular channel)

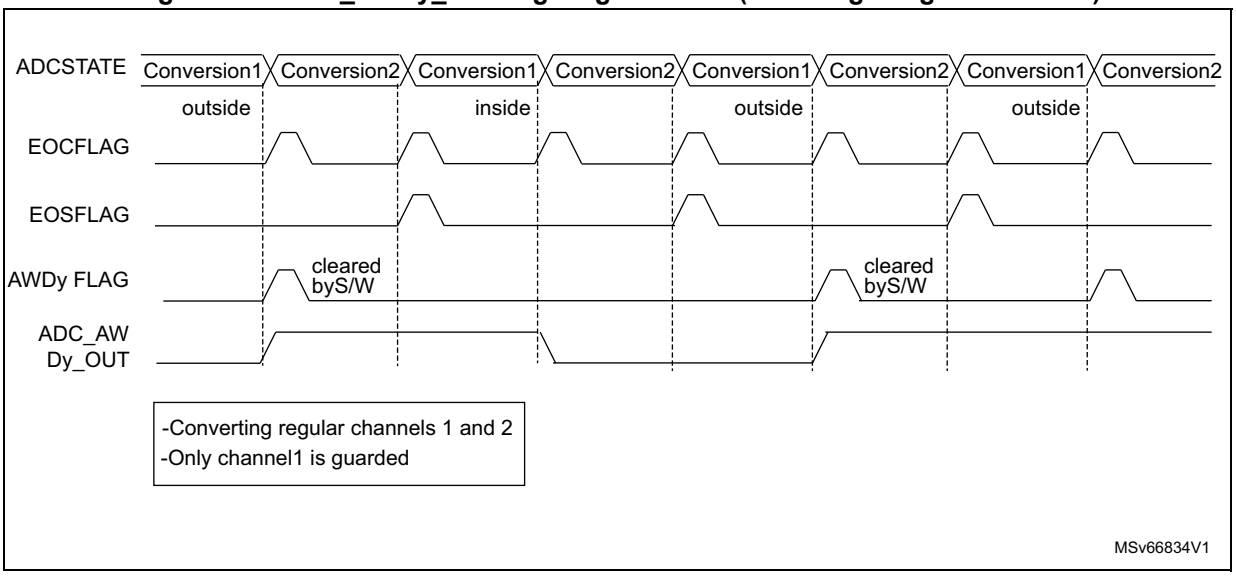
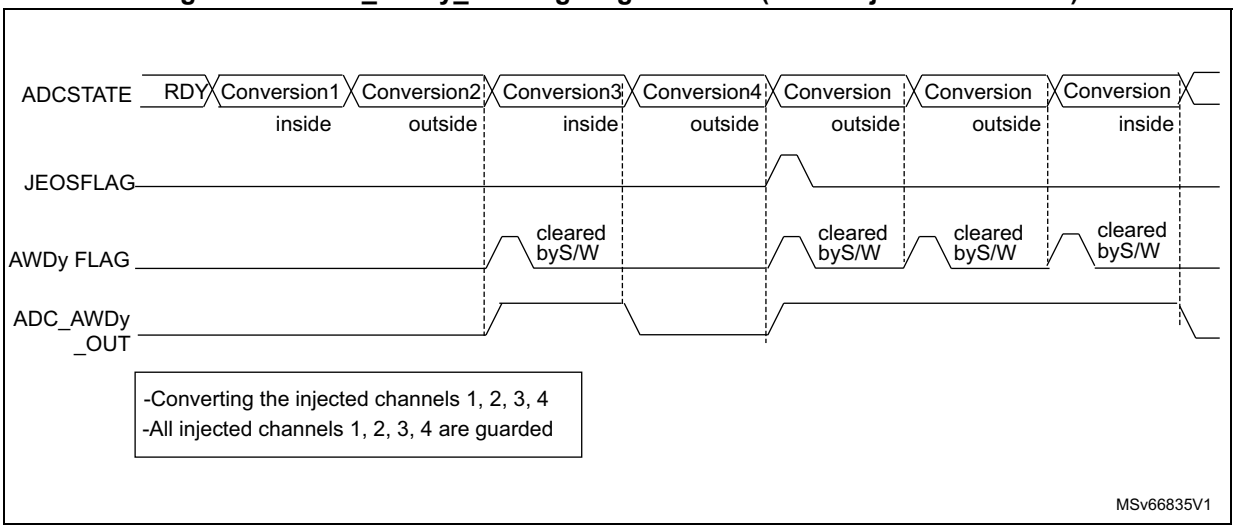


Figure 235. ADC_AWDy_OUT signal generation (on all injected channels)



Analog watchdog threshold control on the fly

LTR[22:0] and HTR[22:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTR[22:0] and HTR[22:0] are updated during the ADC conversion of the ADC guarded channel, resulting in analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new modified threshold, no interrupt and AWDy_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion that started after the threshold update. If AWDy_OUT is already asserted, programming the new thresholds does not deassert the AWDy_OUT signal. However, the watchdog threshold must not be changed twice during the four adc_ker_ck periods.

Analog watchdog filter

With analog watchdog, a valid ADC conversion data range can be configured through the ADC_AWDxLTR and ADC_AWDxHTR registers. When the conversion results have consecutively passed the threshold for more than the watchdog filter value (AWDFILT[2:0] + 1), AWDx_OUT is generated as well as the AWDx flag. Before this value is reached, AWDx output is not activated. The analog watchdog filter can be used only when a single channel is selected for an AWD.

Analog watchdog with gain and offset compensation

When the gain and offset compensation are enabled, the analog watchdog compares the data after the compensation.

26.4.31 Oversampler

The oversampling unit performs data preprocessing to offload the device. It is able to handle multiple conversions and average them into a single data with increased data width, up to 22 bits.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It enables the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement and basic filtering.

The oversampling ratio N is defined by the OVSR[9:0] bits in the ADC_CFGR2 register for regular conversion, or by the JOVSR[3:0] bits in the ADC_CFGR3 register for injected conversion. They can range from 2x to 1024x. The division coefficient *M* consists of a right bit shift up to 10 bits. It is defined through the OVSS[3:0] bits of the ADC_CFGR2 register, for regular conversion or JOVSS[3:0] bits of the ADC_CFGR3 for injected conversion.

The summation unit can yield a result up to 22 bits, which can be left or right shifted at the end. When the right shift is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred to the ADC_DR data register.

In this section, OVSS, LSHIFT, and OVSR are used to explain the functionality; however, the same descriptions also apply to JOVSS, JLSHIFT, and JOVSR for injected conversions.

Figure 236 gives a numerical example of the processing, from a raw 22-bit accumulated data to the final 12-bit result.

Figure 236. 12-bit result oversampling with 10-bit right shift and rounding

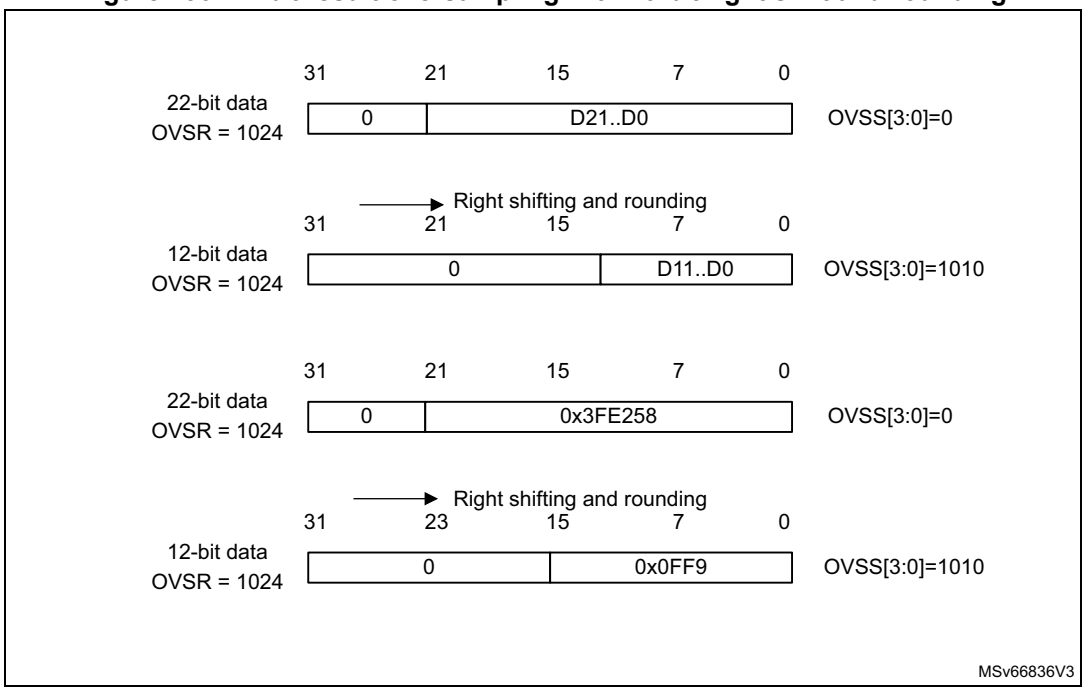


Table 224 gives data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 236. Maximum output results versus N and M

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS =0111	8-bit shift OVSS = 1000
x2	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
x4	0x3FFC	0x3FFC	x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
x16	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
x64	0x3FFC0	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
x256	0xFFF00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF
x1024	0x3FFC00	0x3FFC00	0x1FFE00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC

The conversion timing rate does not change in oversampling mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversion, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase for regular conversion.
- The end of conversion (EOC) occurs once every N conversion, when the oversampled result is available.
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after a total of N x sequence length conversions).

Unlike the conversion rate, the data latency changes in oversampling mode: each single conversion introduces an additional accumulation. As a result, an additional clock cycle is required when all the conversions are complete, after the accumulation and shift.
The EOC flag is set two additional clock cycles after the conversion is complete.

Analog watchdog

The analog watchdog functionality is maintained, with the following differences:

- The RES[1:0] bits are ignored and the comparison is always done using the full 23-bit values HTR[22:0] and LTR[22:0].
- The comparison is performed on the oversampled accumulated value before shifting.

Triggered mode

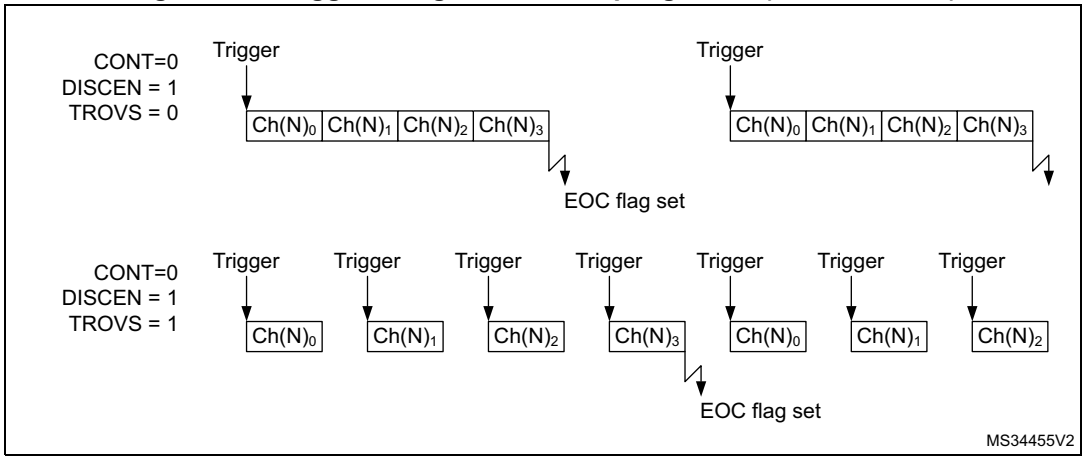
The oversampling can also be used for basic filtering purpose. Although it not a very powerful filter (slow roll-off and limited stop band attenuation), it can act as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched-mode power supply). For this purpose, a specific discontinuous mode can be enabled through the TROVS bit in ADC_CFGR2, to be able to achieve an oversampling frequency defined by the user and independent from the conversion time itself. The TROVS bit is supported with hardware trigger mode.

Figure 237 shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set and OVSR ≠ 0, the content of the DISCEN bit is ignored and considered as 1.

If the TROVS bit is set, neither software trigger nor JAUTO is supported.

Figure 237. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management in oversampling mode

In oversampling mode, injected and regular sequencers can have different behaviors. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously.

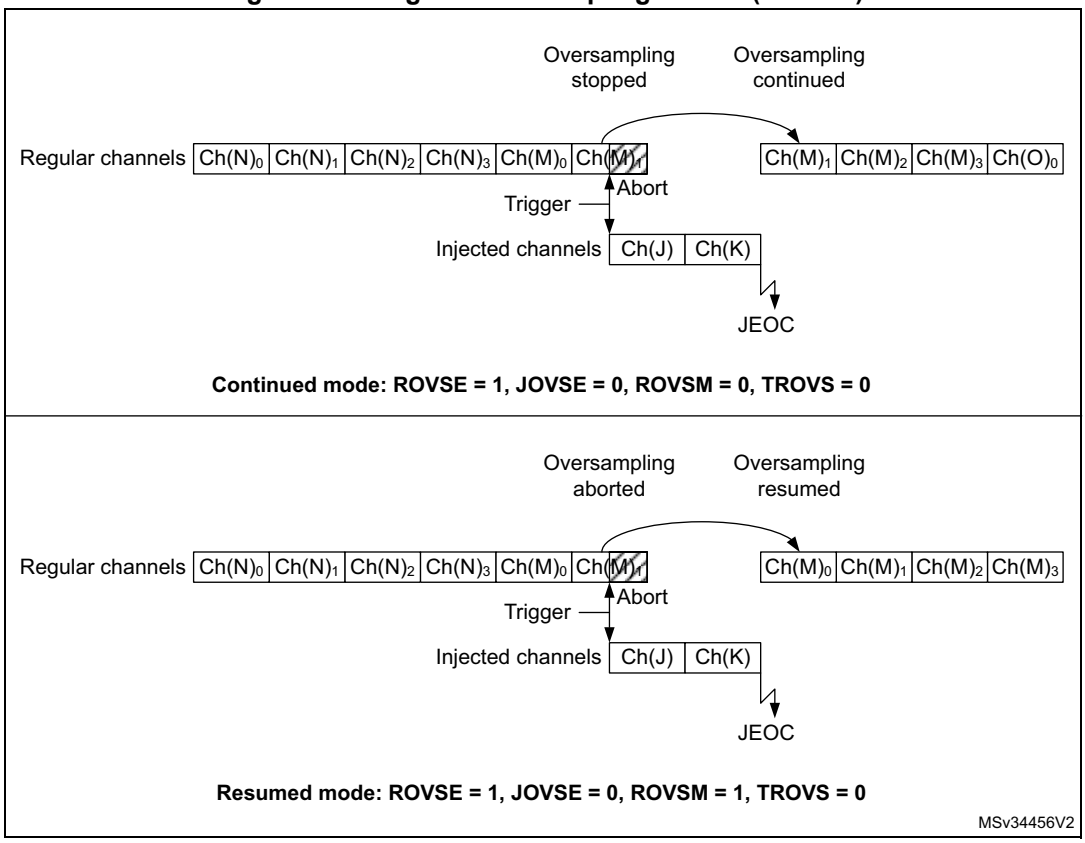
Oversampling regular channels only

The regular oversampling mode ROVSM bit defines how the regular oversampling sequence is resumed if it is interrupted by an injected conversion:

- In continuous mode (ROVSM = 0), the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);
- In resumed mode (ROVSM = 1), the accumulation restarts from 0 (previous conversion results are ignored). This mode guarantees that all the data used for oversampling were converted back-to-back within a single time slot.

Figure 238 gives examples for a 4x oversampling ratio.

Figure 238. Regular oversampling modes (4x ratio)



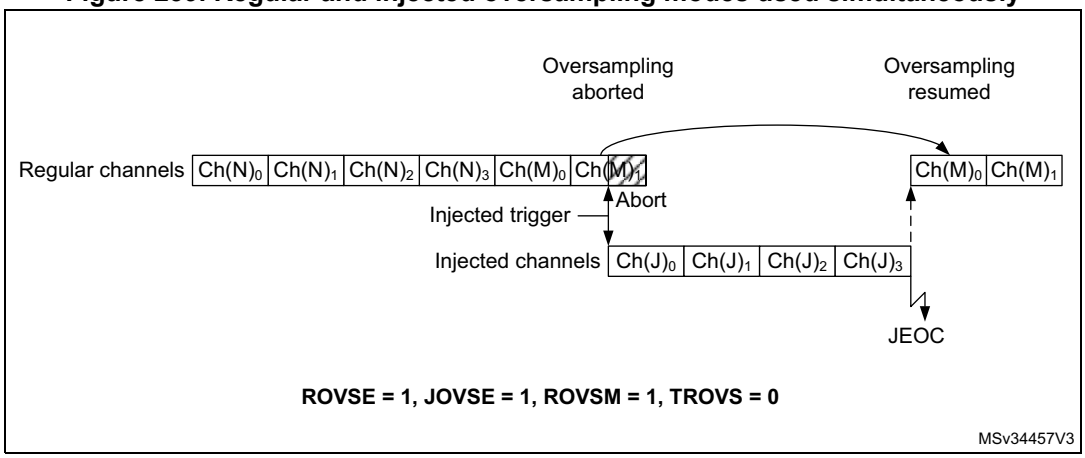
Oversampling injected channels only

The injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling must be resumed (ROVSM bit = 1), as shown in *Figure 239*. This limitation no longer applies in parallel injected oversampling mode.

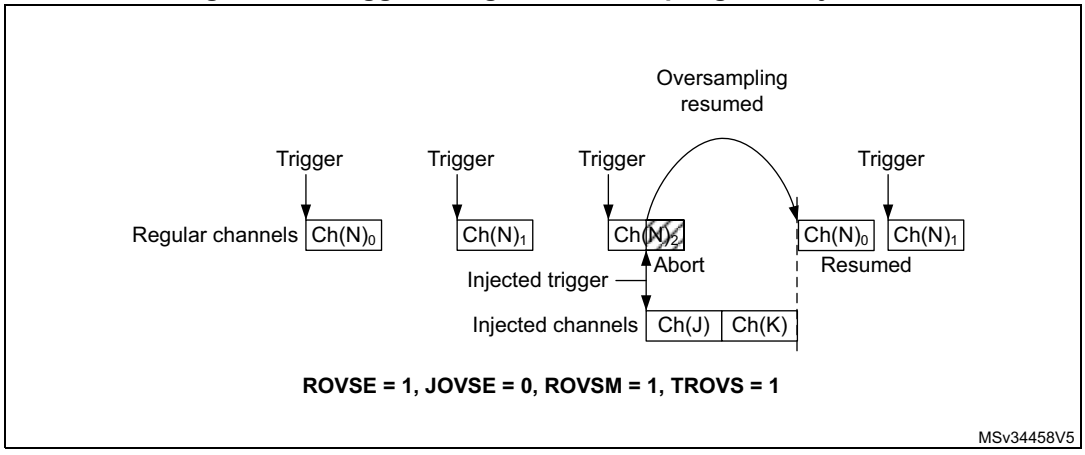
Figure 239. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected oversampling mode must be disabled and the ROVSM bit set (the resumed mode is forced). The JOVSE bit must be reset. The behavior is shown in *Figure 240*.

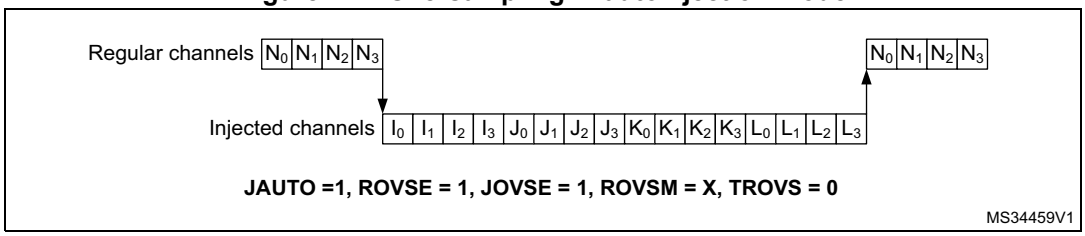
Figure 240. Triggered regular oversampling with injection



Autoinjection mode

It is possible to oversample autoinjected sequences and store all conversion results in registers. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1. Other combinations are not supported. The ROVSM bit is ignored in autoinjection mode. *Figure 241* shows how the conversions are sequenced.

Figure 241. Oversampling in autoinjection mode



Combined modes summary

Table 237 summarizes all combinations, including the modes that are not supported.

Table 237. Oversampler operating mode summary

Regular oversampling ROVSE	Injected oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

26.4.32 Parallel injected oversampling mode

The parallel injected oversampling mode is enabled by setting the JOVSPAR bit of the ADC_CFGR3 register.

To support parallel injected oversampling mode, four dedicated accumulators are implemented for injected conversions, one associated with each JDRy register.

This changes the injected and regular conversion sequences in oversampling mode described in *Section 26.4.31: Oversampler*, because the oversampling can be enabled simultaneously for all injected channels by setting the JOVSPAR bit.

For example, when the oversampling is enabled on two injected channels for scan conversion, the ADC performs the injected conversion (CH1[0], CH2[0], ... CH1[N-1], CH2[N-1]), then each channel is accumulated to ADC_JDR1 and ADC_JDR2 register.

The TJOVS bit of the ADC_CFGR3 register controls each individual conversion:

- When TJOVS is cleared, all conversions in the injected sequence are performed until the oversampling counter is reached.
- When TJOVS is set, each conversion requires an individual trigger.

TJOVS and TJOVSIVE can be used only when JOVSPAR = 1 and TJOVS is compatible with JDISCEN. When the JOVSIVE bit is set in the ADC_CFGR3 register, all intermediate accumulated values stored in the ADC_JDRy registers can be accessed.

When the JADSTP command is executed, the ADC_JDRy registers contain the intermediate accumulated data, and the JOVSCy[9:0] bitfield in the ADC_JOVSCRy register indicates how many accumulations have been performed.

The accumulation counter decrements from the selected ratio value minus one down to 0. Therefore, a value of 0 means that injected oversampling has completed the programmed number of accumulations. The initial counter value is 0, meaning no oversampling has been performed yet.

As long as the intermediate accumulated data count has not reached 0, the oversampling sequence is not complete, and the conversion data available in JDRy corresponds to the raw accumulated value without any right bit shift. The right bit shift is applied only once the oversampling sequence is fully completed.

If for any reason the injected oversampling needs to be stopped, the software can retrieve the intermediate data and the accumulation count, and calculate the data based on oversampling data divided by the accumulation counter.

Figure 242. Triggered injected parallel oversampling for 3 channels example

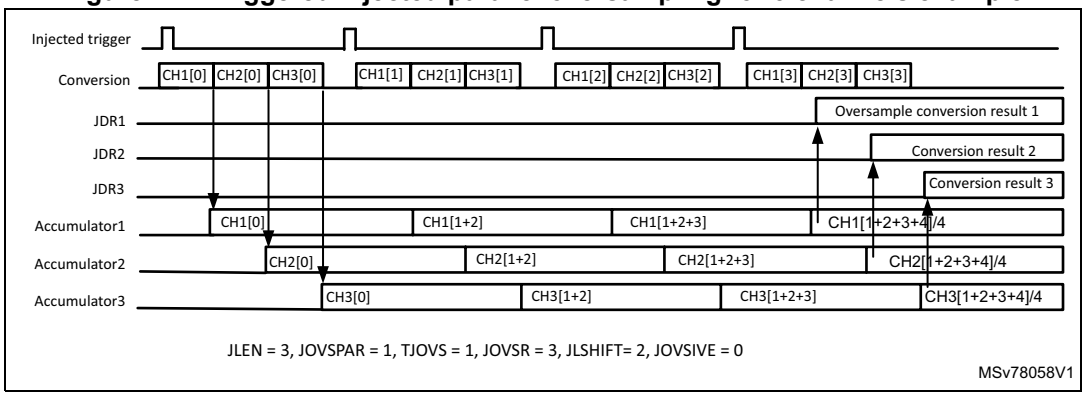


Figure 243. Injected oversampling with intermediate value enabled example

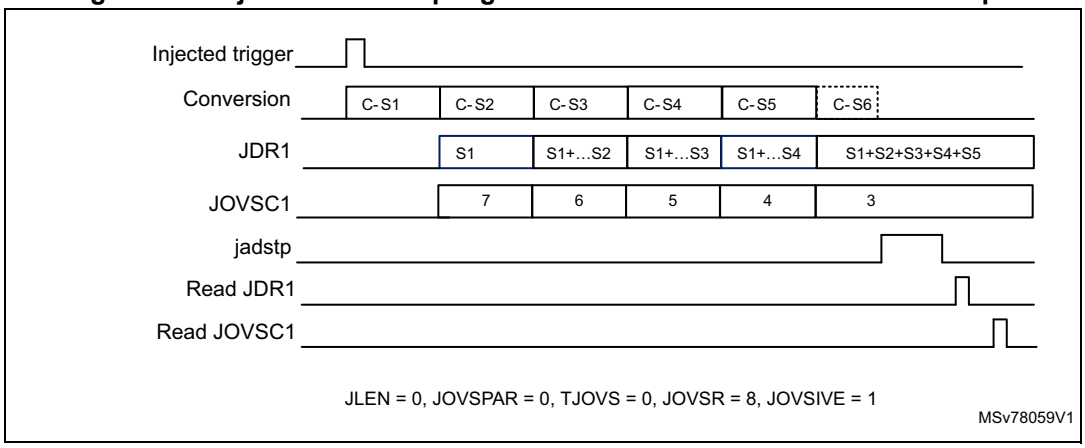
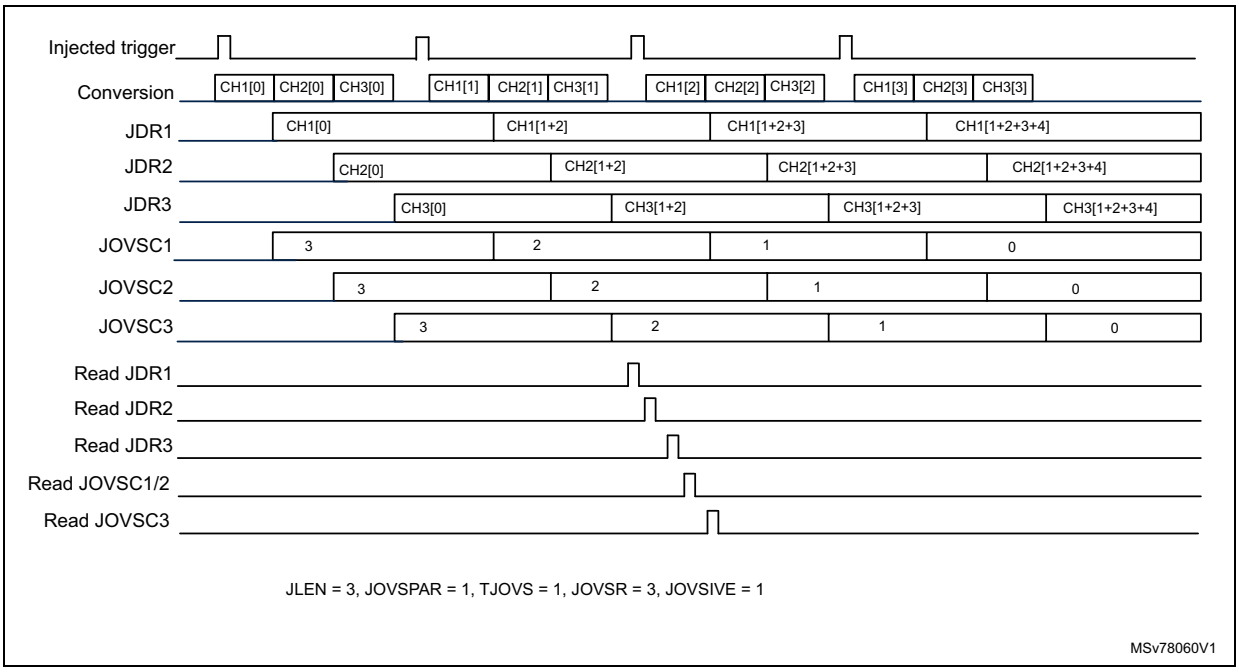


Figure 244. Triggered injected parallel oversampling for 3 channels with accumulated data example



26.4.33 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 245*).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADC master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + regular simultaneous mode
- Regular simultaneous mode + alternate trigger mode
- Injected simultaneous mode + interleaved mode

In dual ADC mode (when the DUAL[4:0] bits in the ADCC_CCR register are not equal to zero), the following bits are shared between the master and slave ADCs, with the slave ADC bits being ignored and always applying to the corresponding bits of the master ADC:

- ADC_CFGR1 register: DMNGT[2:0], RES[1:0], EXTEN[1:0], OVRMOD, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JAUTO, JDISCEN.
- ADC_CFGR2 register: ROVSE, JOVSE, TROVS, ROVSM, BULB, SWTRIG, SMPTRIG, OVSR[9:0].
- ADC_GCOMP register: GCOMP.
- ADC_CR register: ADSTART, ADSTP, JADSTART, JADSTP.
- ADC_CFGR1 register: CONT, DISCEN, DISCNUM[2:0].
- ADC_SQR1 register: LEN (for the regular conversion).
- ADC_JSQR register: JLEN[1:0], JEXTSEL[4:0], JEXTEN[1:0] (for the injected conversion).
- ADC_CFGR3 register: JOVSPAR, JOVSE, JOVSIVE, TJOVS, JOVSR[9:0] (for the injected conversion).

For dual mode usage, the sampling time of the master and slave channels must be equal.

To start a conversion in dual mode, the user must program the EXTEN[1:0], EXTSEL, JEXTEN[1:0] and JEXTSEL bits of the master ADC only to configure a software or hardware trigger and a regular or injected trigger. The EXTEN[1:0], JEXTEN[1:0], and JEXTSEL bits of the slave ADC are don’t care).

In regular simultaneous or interleaved modes, once the user sets the ADSTART or ADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, ADSTART or ADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

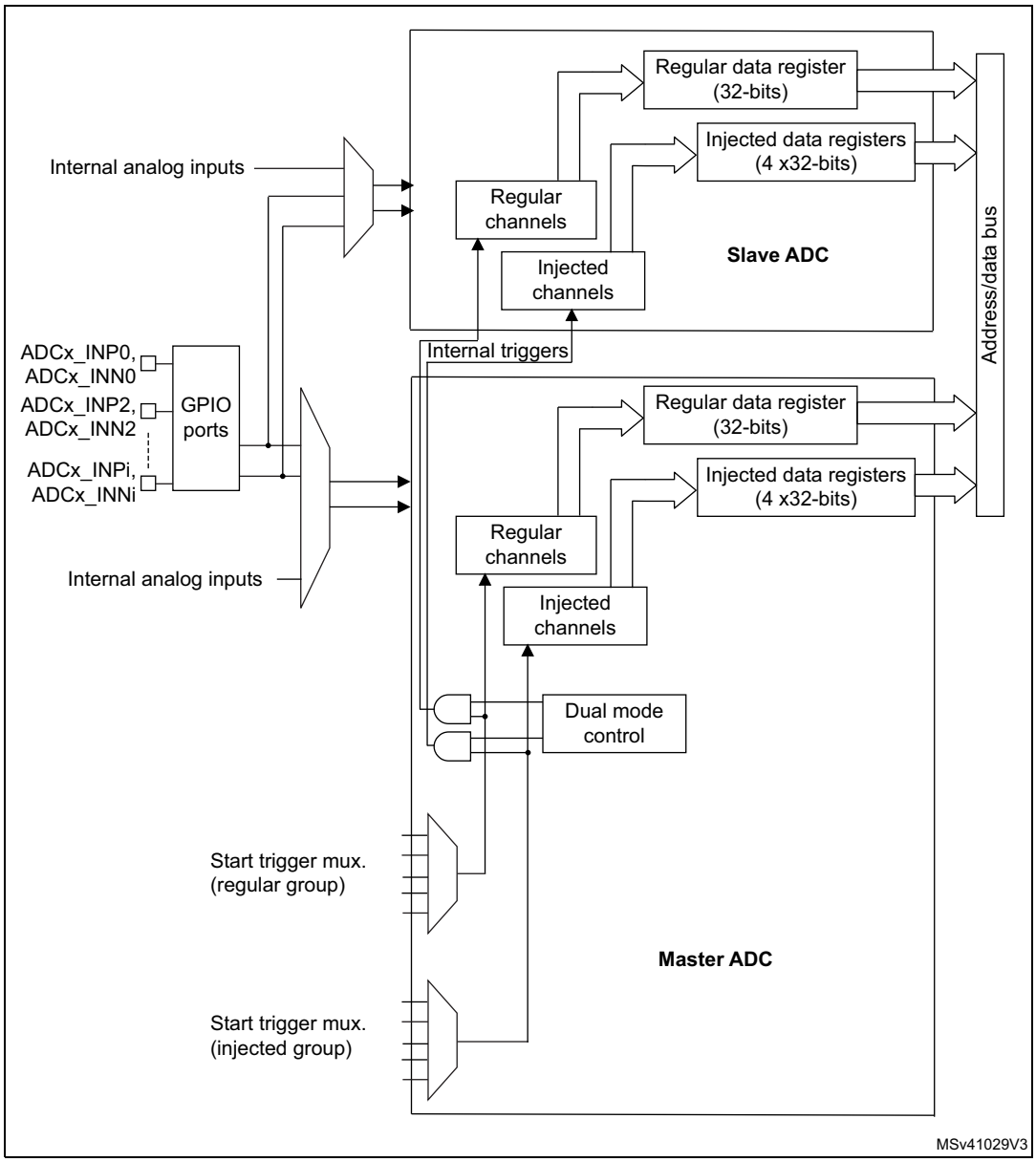
In injected simultaneous or alternate trigger modes, once the user sets the JADSTART or JADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, JADSTART or JADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCC_CDR). The flags can also be read in parallel by reading the dual-mode status register (ADCC_CSR).

Note: In dual mode, BULB, JAUTO, SMPTRIG, and the injected queue are not supported.

During ADC calibration, the DUAL[4:0] bits must be set to 00000.

Figure 245. Dual ADC block diagram



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCC_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode with independent regular conversion

The injected simultaneous mode is selected by programming DUAL[4:0] bits to 0b00101.

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

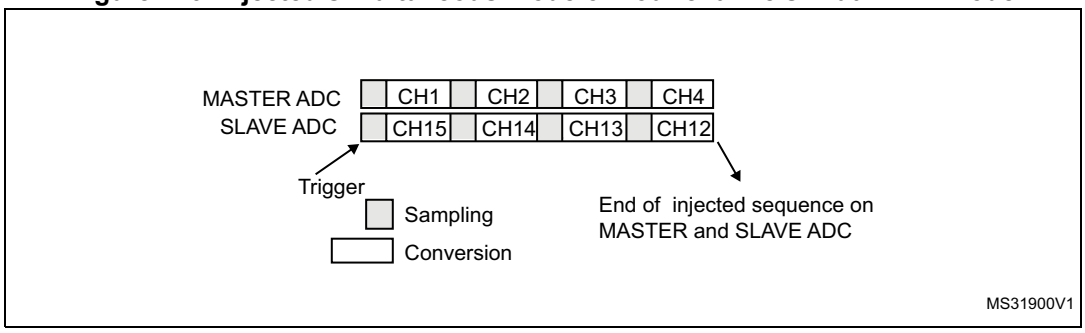
In simultaneous mode, the user software must convert sequences with the same length. Sampling times for channels with the same sequence number must be equal.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

The software is notified by interrupts when it can read the data:

- If the duration of the master-injected sequence is equal to the duration of the slave injected one (as shown in *Figure 246*), the software can enable only one of the two JEOS interrupts (for example the master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 246. Injected simultaneous mode on four channels: Dual ADC mode



If JDISCEN is set, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with the AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the ongoing regular sequence and the associated delay phases are ignored.
  There is the same behavior for regular sequences occurring on the slave ADC.

In this mode, the following register bits are shared between the master and slave ADCs:

JADSTART, JADSTP, JDISCEN, JOVSE, JLEN, JEXTEN, JEXTSEL, and GCOMPEN, JOVSPAR, JOVSIVE, JOVSE, and TJOVS.

Regular simultaneous mode with independent injected conversions

The regular simultaneous mode is selected by programming DUAL[4:0] bits to 0b00110.

This mode applies to a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR1 register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on the master or on the slave) aborts the current simultaneous conversions both for master and slave. They are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, the user software must convert sequences with the same length. Sampling times for channels with the same sequence number must be equal.

The software is notified by interrupts when it can read the data:

- As the duration of the master regular sequence is equal to the duration of the slave one (as shown in *Figure 247*), the software can enable only one of the two EOC interrupts (for example the master EOC) and read both converted data from the common data register (ADCC_CDR).
  The user can read data from either ADCC_CDR or ADCC_CDR2. Once software selects one of these registers, it must continue using the same register until the ADSTP procedure is executed.

The regular data can also be read using the DMA. Several methods are possible:

- Use two DMA channels (one for the master and one for the slave). In this case DAMDF[1:0] bits must be kept cleared:
  a) Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  b) Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Configure the dual ADC mode data format DAMDF[1:0] bits, which leaves one DMA channel free for other uses:
  a) Set DAMDF[1:0] = 0b10 or 0b11 (depending on resolution).
  b) A single DMA channel is used (the one corresponding to the master). Configure the DMA master ADC channel to read the common ADC data register (ADCC_CDR or ADCC_CDR2).
  c) A single DMA request is generated each time both the master and slave EOC events occur. At that moment:
  – The slave ADC converted data is available in the upper half-word of the 32-bit ADCC_CDR register
  – The master ADC converted data is available in the lower half-word of the ADCC_CDR register.

Alternatively, data can be read from ADCC_CDR2:

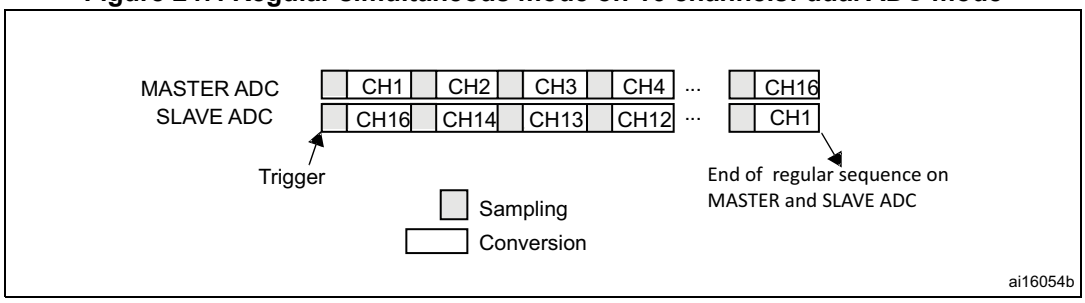
– The first read returns the master ADC data.

– The second read returns the slave ADC data.
After these reads, a DMA request is generated. The DMA transfers the slave data from the same data register and handles two transfers for dual mode operation.

d) Both EOC flags are cleared when the DMA reads the ADCC_CDR register.

Note: When DAMDF[1:0] = 0b10 or 0b11, the user must program the same number of conversions in the master and in the slave sequence. Otherwise, the remaining conversions do not generate a DMA request.

Figure 247. Regular simultaneous mode on 16 channels: dual ADC mode



If the DISCEN bit is set, each "*n*" simultaneous conversion of the regular sequence require a regular trigger event to occur ("n" is defined by DISCNUM).

This mode can be combined with the AUTDLY mode:

- Once a simultaneous conversion of the sequence has completed, the next conversion in the sequence starts only after the EOC flag of both the master and the slave are cleared.
- Once a simultaneous regular sequence of conversions has completed, a new regular trigger event is accepted only after the EOC flag of both the master and the slave are cleared. Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

The DMA can be used to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits DAMDF must be set to 0b10 or 0b11.

Interleaved mode with independent injected conversions

This mode is selected by programming DUAL[4:0] bits to 0b00111.

It can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately,
- The slave ADC starts after a delay of several ADC clock cycles after the sampling phase of the master ADC has completed (during the master ADC conversion period).

The minimum delay between two conversions in interleaved mode is configured in the DELAY bits in the ADCC_CCR register (see *Table DELAY bits versus ADC resolution*). This delay starts counting one half cycle after the end of the sampling phase of the master

conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However, the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

As the CONT bit is set on both the master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the master or slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Table 238. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$
0001	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$
0010	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$
0011	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$
0100	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$
0101	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0110	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
0111	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1000	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1001	10 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1010	11 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1011	12 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
1100	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$
others	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$

Figure 248. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode

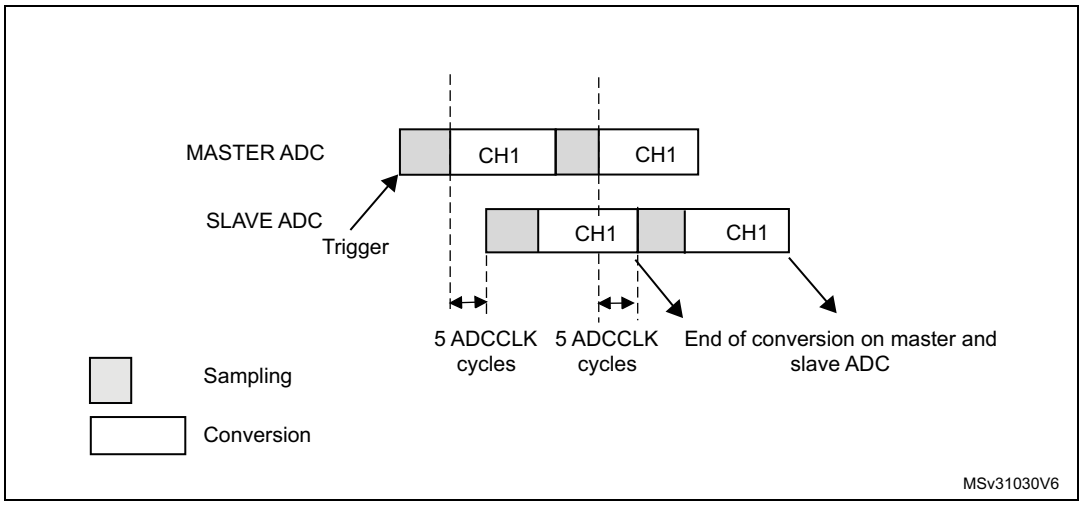
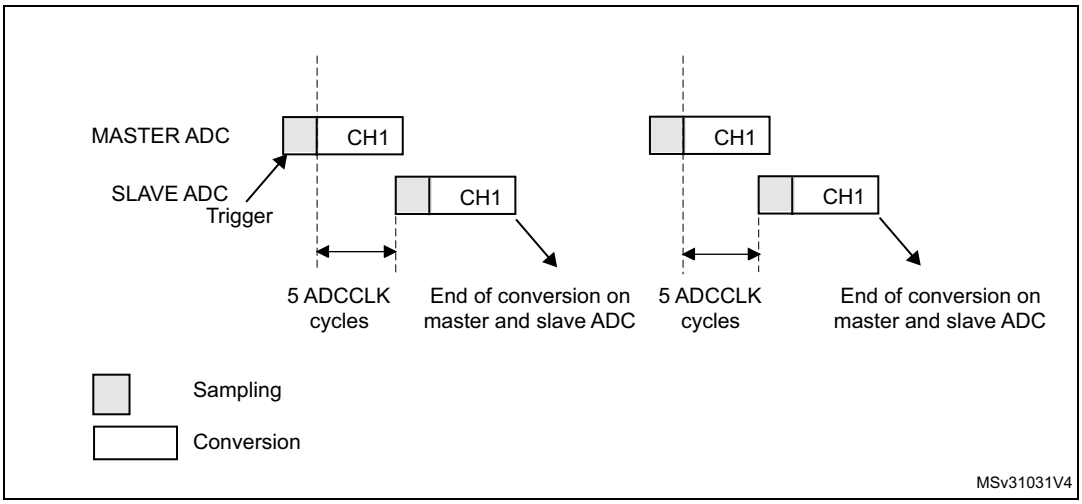


Figure 249. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode



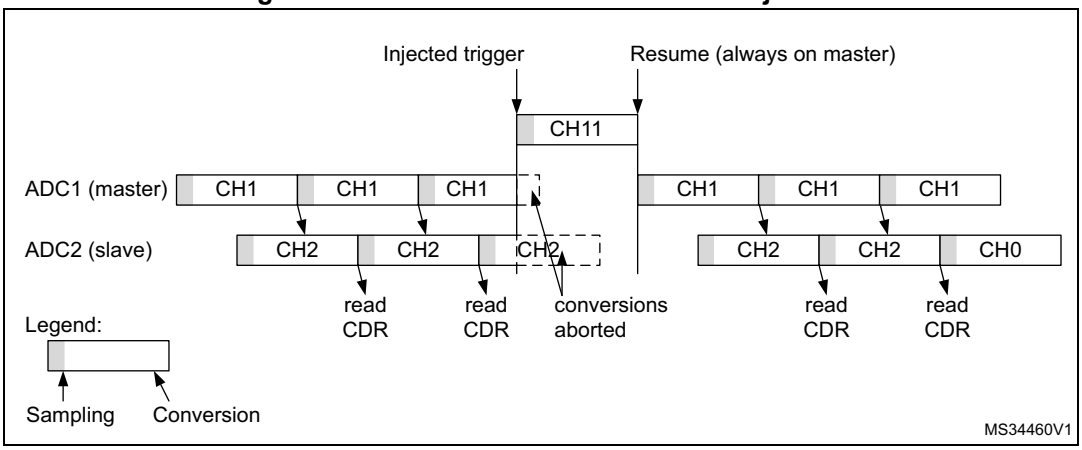
If the DISCEN bit is set, each "*n*" simultaneous conversion ("*n*" is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In interleaved mode, injected conversions are supported. When injection is done (either on master or slave), both master and the slave regular conversions are aborted and the sequence is restarted from the master (see *Figure 250*).

Note: In interleaved mode, the ADCx_AWDy_OUT signal for the master ADC is synchronized with the EOC of the master ADC, but the interrupt flag is synchronized with the EOC/JEOC flag of the slave ADC.

In addition, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 250. Interleaved conversion with injection



Alternate trigger mode with independent regular conversions

The alternate trigger mode is selected by programming DUAL[4:0] bits to 0b01001.

This mode can only be started on an injected group. The source of the external trigger comes from the multiplexer injected group of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN[1:0] must not be 0x0.

Injected discontinuous mode disabled (JDISCEN = 0)

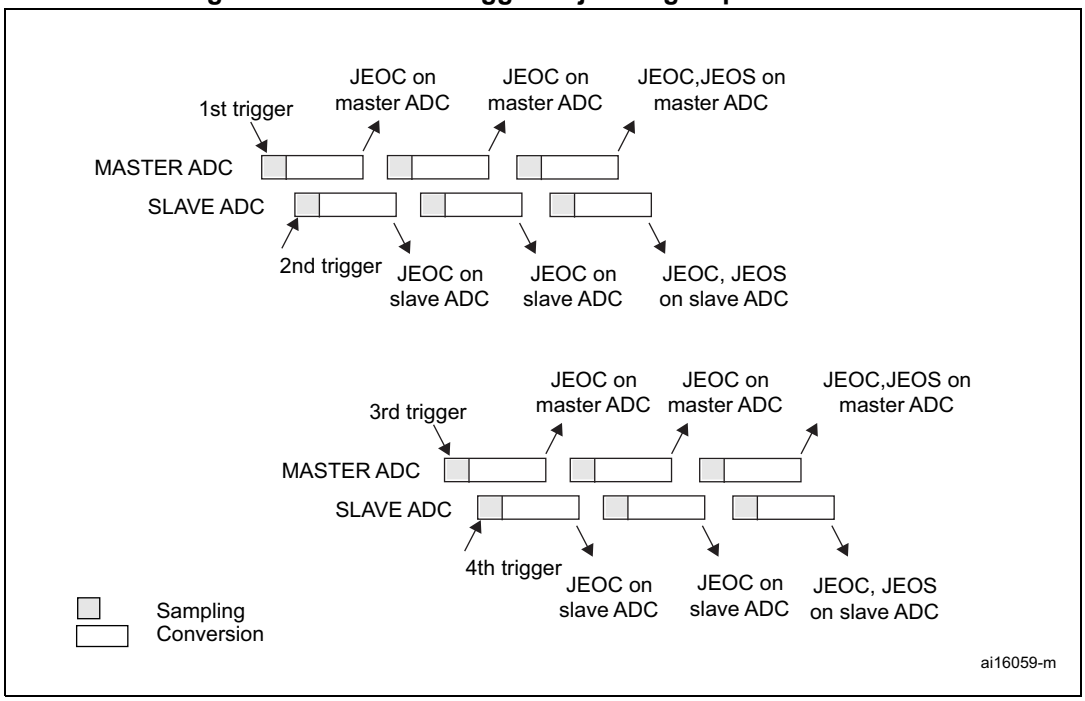
1. When the first trigger occurs, all injected master ADC channels in the group are converted.
2. When the second trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on...

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

Figure 251. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case, the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between two trigger events must be greater than or equal to one ADC clock period. The minimum time interval between two trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN = 1)

If the injected discontinuous mode is enabled for both master and slave ADCs:

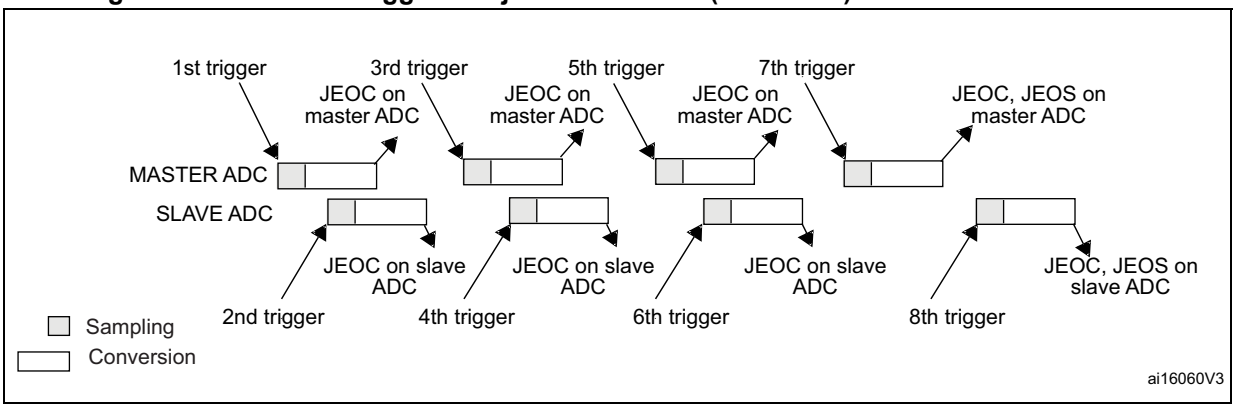
1. When the first trigger occurs, the first injected channel of the master ADC is converted.
2. When the second trigger occurs, the first injected channel of the slave ADC is converted.
3. And so on...

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

Figure 252. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming DUAL[4:0] bits to 0b00001.

The simultaneous conversion of a regular group can be interrupted to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, the user software must convert sequences with the same length.

Combined regular simultaneous and alternate trigger mode

This mode is selected by programming DUAL[4:0] bits to 0b00010.

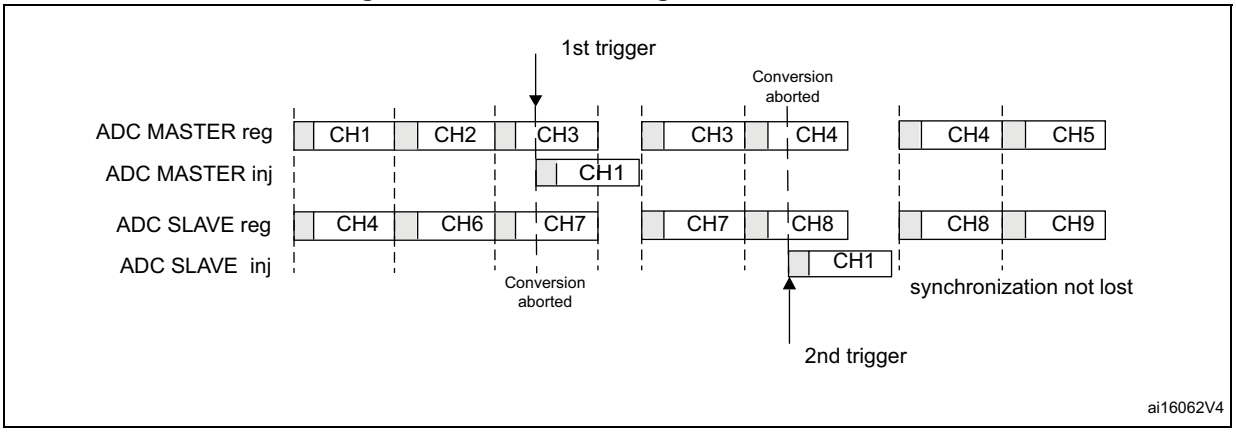
The simultaneous conversion of a regular group can be interrupted to start the alternate trigger conversion of an injected group. *Figure 253* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion immediately starts after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

Note: In combined regular simultaneous + alternate trigger mode, the user software must convert sequences with the same length.

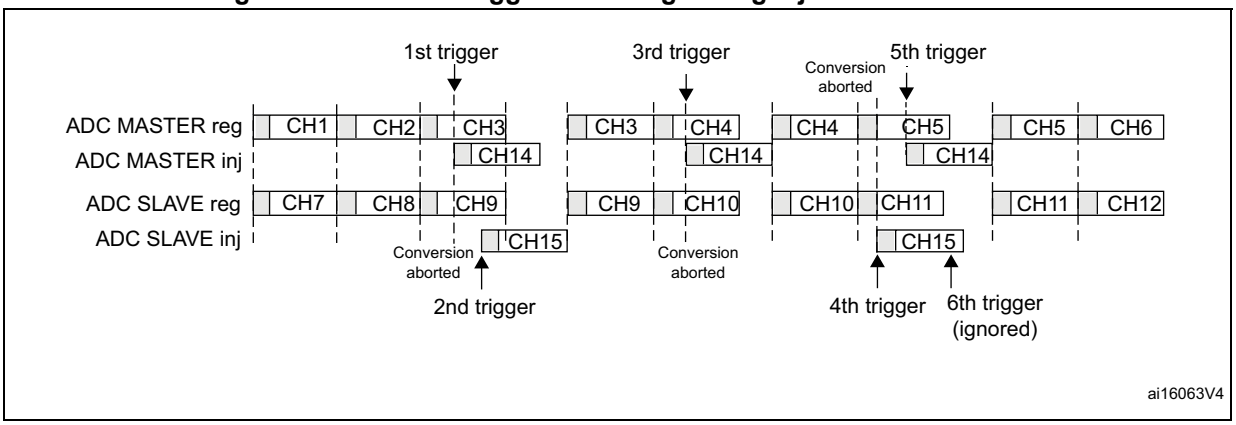
The software trigger is not supported for injected conversions.

Figure 253. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 254* shows the behavior in this case (note that the sixth trigger is ignored because the associated alternate conversion is not complete).

Figure 254. Case of trigger occurring during injected conversion



Combined injected simultaneous and interleaved mode

This mode is selected by programming DUAL[4:0] bits to 0b00011.

An interleaved conversion can be interrupted with a simultaneous injected event.

In this case, the interleaved conversion is immediately interrupted and the simultaneous injected conversion starts. At the end of the injected sequence, the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master's one. *Figure 255*, *Figure 256*, and *Figure 257* show examples of this behavior.

Note: In this mode, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 255. Interleaved single channel CH0 with injected sequence CH11, CH12

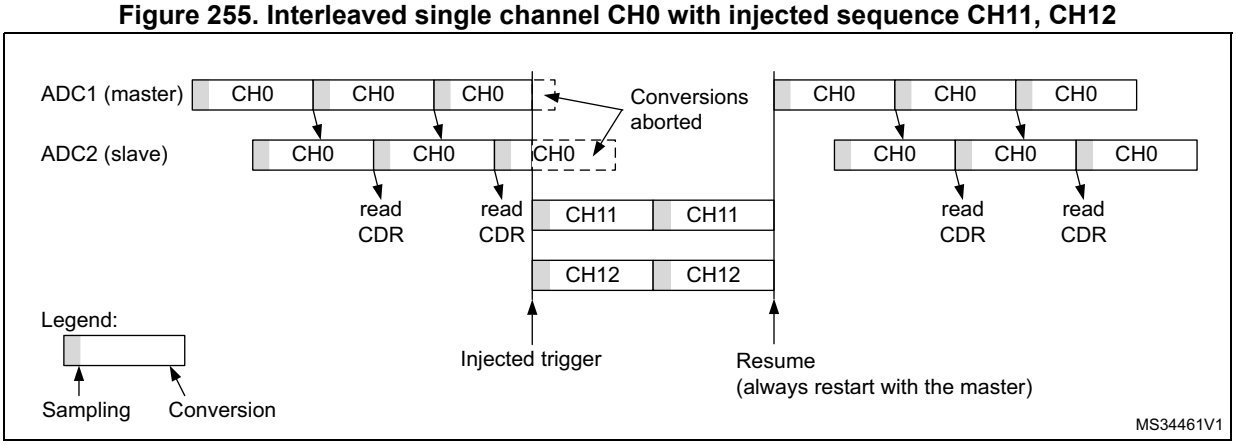


Figure 256. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 case 1: Master interrupted first

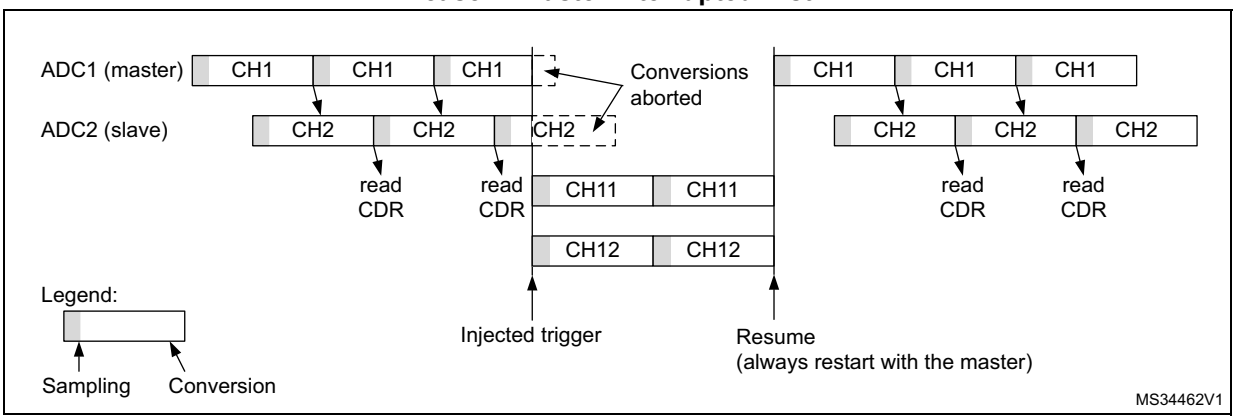
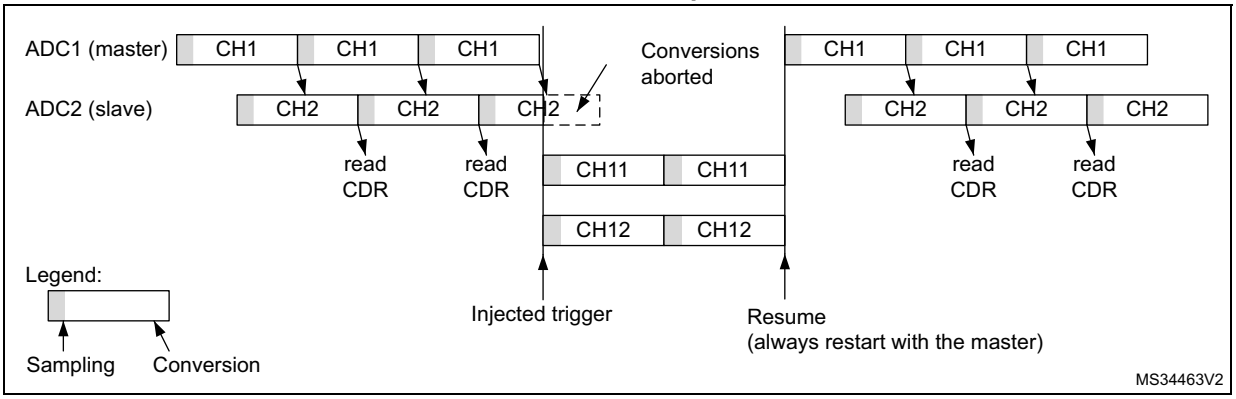


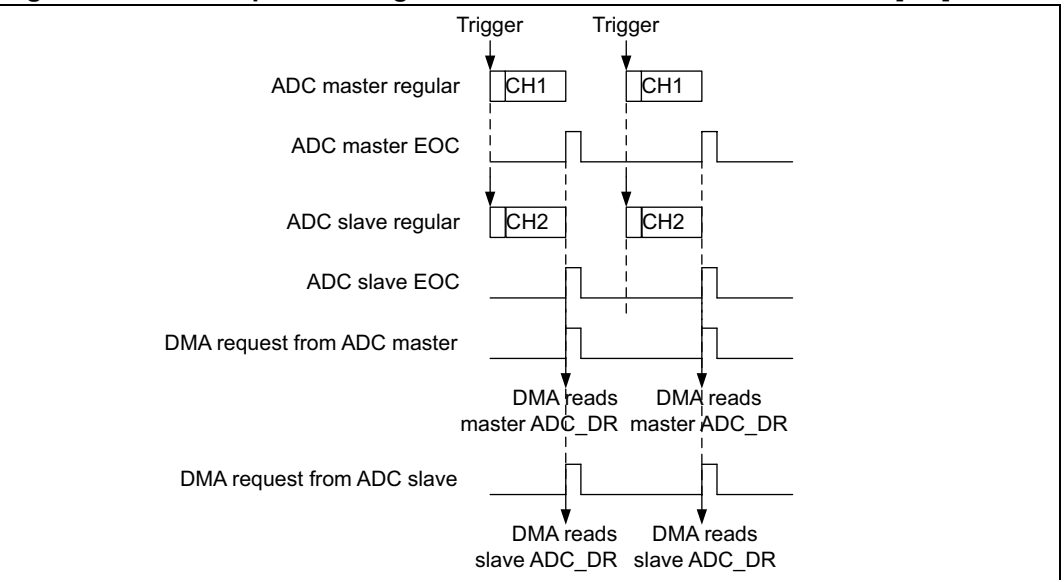
Figure 257. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 258: DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00*).

Figure 258. DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00



Configuration where each sequence contains only one conversion. Master and slave timings are equal.

MSv31032V3

In interleaved mode or in simultaneous regular mode, it is also possible to save one DMA channel and transfer both data using a single DMA channel. To do this, the DAMDF[1:0] bits must be configured in the ADCC_CCR register:

- **DAMDF[1:0] = 0b10, 32-bit format:**

  A DMA request is generated alternatively after the master and slave EOC events have occurred. At that time, the data are alternatively available in the 32-bit register ADCC_CDR2 32-bit register.

  This mode is used when the data width is above 16 bits (conversion data width can exceed 16 bits when functions such as oversampling, compensation and left-bit shift, are used).

  Behavior: A DMA request is generated each time a new 32-bit data is available:

  – First DMA request: ADCC_CDR2[31:0] = MST_ADC_DR[31:0]

  – Second DMA request: ADCC_CDR2[31:0] = SLV_ADC_DR[31:0]

- **DAMDF[1:0] = 0b10, 16-bit format**:

  A DMA request is generated alternatively after the master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCC_CDR contains the two half-words representing two ADC-converted data. The slave ADC data takes the upper half-word and the master ADC data takes the lower half-word. DMA transfers two data by one request.

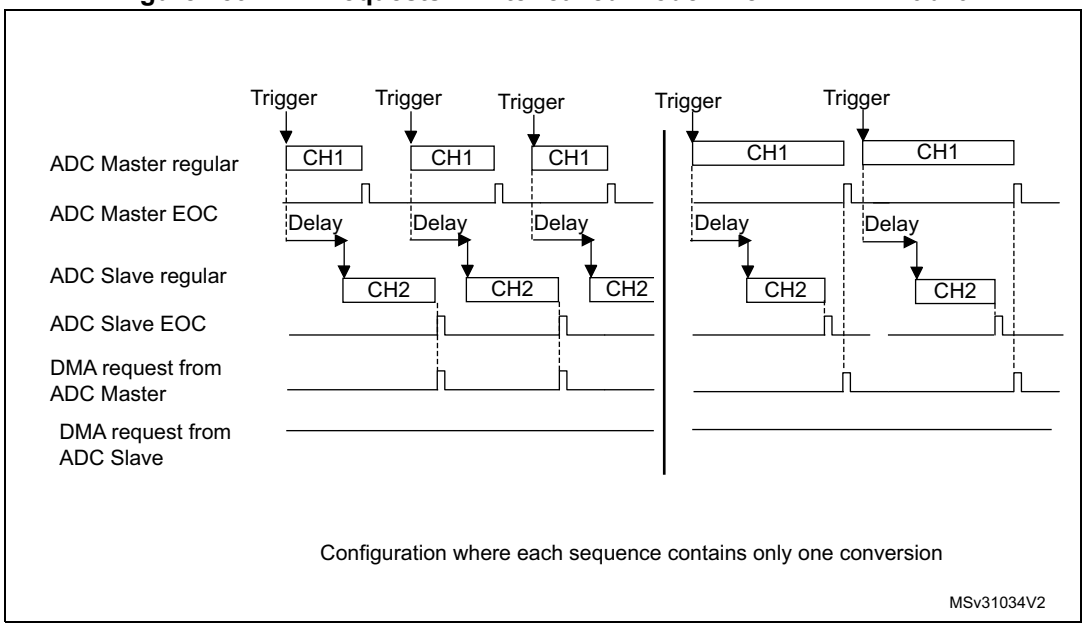
Any value above 16-bit in the master or the slave-converted data is truncated to the least 16 significant bits.

Behavior: A DMA request is generated each time a new 32-bit data is available:

– First DMA request:
ADCC_CDR[31:0] = (SLV_ADC_DR[15:0] << 16) | MST_ADC_DR[15:0]

– Second DMA request:
ADCC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 259. DMA requests in interleaved mode when DAMDF = 0b10



Note: When using dual ADC mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion data is available.

- **DAMDF[1:0] = 0b11**:

  This mode is similar to the DAMDF[1:0] = 0b10 configuration. The only difference is that on each DMA request (two data are available), two bytes representing two ADC converted data are transferred as a half-word.

  This mode is used when the resolution is 6 bits or 8 bits and data are not signed.

  Behavior: A DMA request is generated each time two data items are available:

  – First DMA request:
  ADCC_CDR[15:0] = (SLV_ADC_DR[7:0] << 8)| MST_ADC_DR[7:0]

  – Second DMA request:
  ADCC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

Note: DMA burst mode is not supported in ADC dual mode.

Overrun detection when OVRMOD = 0

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the DAMDF configuration). The EOC bit

corresponding to one ADC might remains set because the data register of this ADC contains valid data.

Overrun detection when OVRMOD = 1

In dual ADC mode, if an overrun occurs on either ADC, the data register is overwritten, causing continuous DMA requests. This results in loss of data coherency.

DMA one-shot mode/ DMA circular mode when multi-ADC mode is selected

When DAMDF mode is selected (0b10 or 0b11), DMNGT[1:0] bits of the ADCC_CCR register must also be configured to select between DMA one-shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA*

Stopping the conversions in dual ADC modes

In dual ADC mode, the user software must set the ADSTP/JADSTP control bits in the master ADC to stop the conversions of both ADC. The other ADSTP control bit on the slave ADC has no effect in dual ADC mode.

Once both ADCs are effectively stopped, the ADSTART/JADSTART bits on the master and slave ADCs are both deasserted by hardware.

Analog watchdog in dual mode

The analog watchdog is supported in all dual modes.

Disabling the ADC in dual mode

During a dual-mode operation, the master ADC shares the ADC clock with the slave ADC. To disable the ADC correctly through the ADDIS bit of ADC_CR, set the ADSTP (or JADSP) bit on the master ADC. Once ADSTP (or JADSTP) becomes 0 both for the master and slave ADCs, set the ADDIS bit on the master ADC, then set the ADDIS bit on the slave ADC.

Autoinjection mode in dual mode

Autoinjection mode is not supported when DUAL[4:0] equals 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, or 0b01001.

Additionally, parallel injected oversampling mode is not supported in autoinjection mode.

Dual ADC modes supported in oversampling mode

It is possible to have oversampling enabled when working in dual ADC configuration. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (that means when ROVSE = 1 or JOVSE = 1).

26.4.34 Temperature sensor

The temperature sensor can be used to measure the junction temperature (T_J) of the device.

The temperature sensor is internally connected to the ADC input channels, which are used to convert the sensor output voltage to a digital value (see *Table: ADC interconnection* in *Section 26.4.2: ADC pins and internal signals* for more details). When not in use, the sensor can be put in power-down mode. It supports the temperature range specified in the product datasheet.

Figure 260 shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

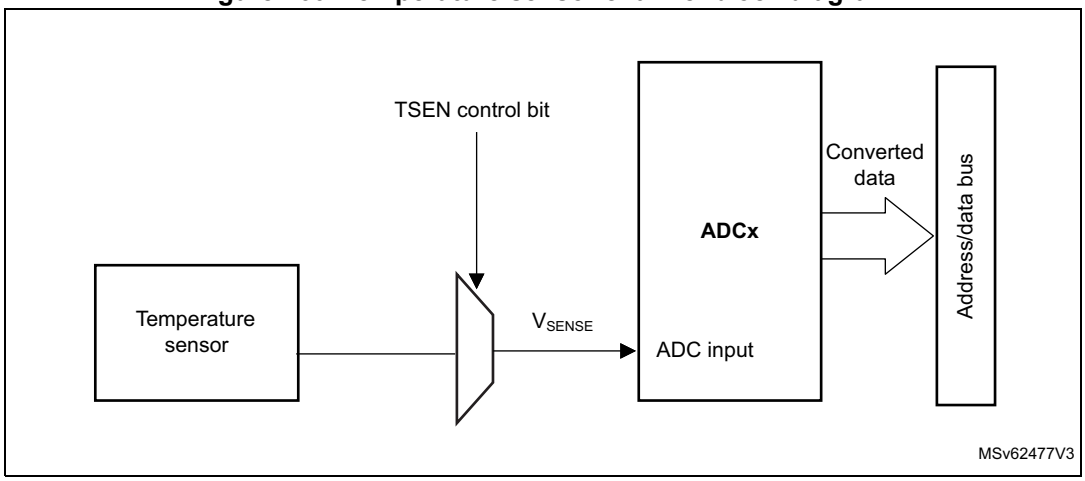
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel, which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 260 shows the block diagram of the temperature sensor.

Figure 260. Temperature sensor channel block diagram



Reading the temperature

To use the sensor:

1. Select the ADC input channel that is connected to V_{SENSE}.
2. Program with the appropriate sampling time (refer to the electrical characteristics section of the device datasheet).
3. Set the TSEN bit in the ADCC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{SENSE} data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in °C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Caution: The above formula is valid only if the V_{REF+} condition specified in the table *Temperature sensor calibration values* of the datasheet are respected.

Note: The sensor has a startup time after waking from power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN, and TSEN bits must be set simultaneously.

26.4.35 Monitoring the internal voltage reference

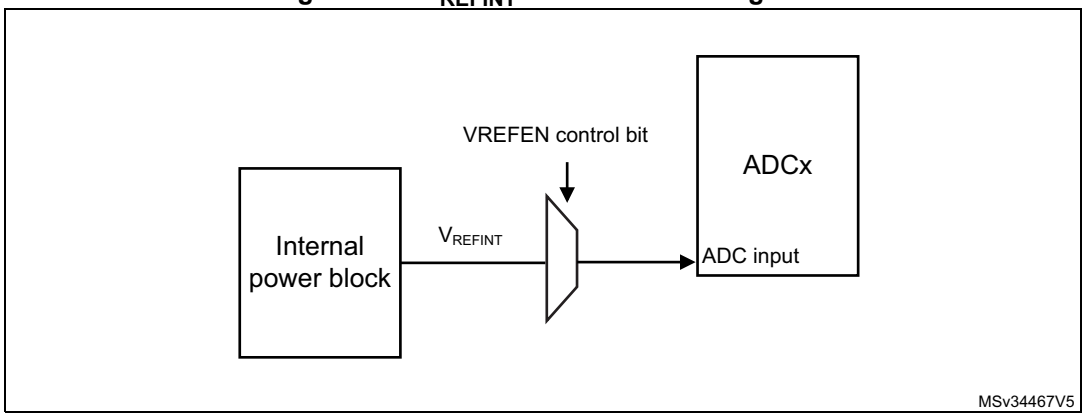
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to the *ADC interconnection* table in *Section 26.4.2: ADC pins and internal signals* for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 261 shows the block diagram of the V_{REFINT} sensing feature.

Figure 261. V_{REFINT} channel block diagram



1. The VREFEN bit of the ADCC_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{REF+} can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value.
- VREFINT_DATA is the actual VREFINT output value converted by ADC.

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For some applications, the V_{REF+} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{REF+}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times VREFINT_CAL \times ADC_DATA}{VREFINT_DATA \times FULL_SCALE}$$

For applications where V_{REF+} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

Where:

– VREFINT_CAL is the VREFINT calibration value.
– ADC_DATA is the value measured by the ADC on channel x (right-aligned).
– VREFINT_DATA is the actual VREFINT output value converted by the ADC.
– FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, we obtain $2^{12} – 1 = 4095$, or $2^8 – 1 = 255$ with 8-bit resolution.

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

26.4.36 Monitoring the OPAMP output voltage

The ADC is connected to a OPAMP internal channel. To measure the OPAMP output voltage, enable the OPAMP internal channel through OPAMP registers.

26.4.37 Monitoring the supply voltage

The ADCs are connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.

26.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (ADRDY flag)
- On the end of any conversion for regular groups (EOC flag)
- On the end of a sequence of conversion for regular groups (EOS flag)
- On the end of any conversion for injected groups (JEOC flag)
- On the end of a sequence of conversion for injected groups (JEOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2, and AWD3 flags)
- When the end of the sampling phase occurs (EOSMP flag)
- When the data overrun occurs (OVR flag)
- When the ADC internal voltage regulator output is ready (LDORDY flag)
- When the injected sequence context queue overflows (JQOVF flag)

Separate interrupt enable bits are available for flexibility.

Note: Interrupts are generated by the AHB clock domain. Since the ADC clock is adc_ker_ck, interrupt flags are delayed due to the synchronization between adc_ker_ck and AHB clock domains.

Table 239. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No
	End of conversion of a regular group	EOC	EOCIE			
	End of conversion sequence of a regular group	EOS	EOSIE			
	End of conversion of an injected group	JEOC	JEOCIE			
	End of conversion sequence of an injected group	JEOS	JEOSIE			
	Analog watchdog 1 flag is set	AWD1	AWD1IE			
	Analog watchdog 2 flag is set	AWD2	AWD2IE			
	Analog watchdog 3 flag is set	AWD3	AWD3IE			
	End of sampling phase	EOSMP	EOSMPIE			
	Overrun	OVR	OVRIE			
	ADC internal voltage regulator ready	LDORDY	LDORDYIE			
	Injected context queue overflows	JQOVF	JQOVFIE			

26.6 ADC registers (for each ADC)

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

26.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDY	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
			rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: ADC internal voltage regulator output ready flag

This bit is set by hardware and cleared by software. It indicates that the ADC internal voltage regulator output is ready and that the ADC can be enabled or calibrated.

0: ADC LDO internal voltage regulator disabled

1: ADC LDO internal voltage regulator enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an overflow of the injected queue of context occurs. It is cleared by software writing 1 to it or when ADEN = 0.

Refer to *Section 26.4.23: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD3LTR and ADC_AWD3HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD2LTR and ADC_AWD2HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD1LTR and ADC_AWD1HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it or when ADEN = 0.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the corresponding ADC_JDRy register.

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel (a new conversion has completed while the EOC flag was already set) or when adc_hclk clock is too slow to manage the data.

It is cleared by software writing 1 to it or when ADEN = 0.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it or when ADEN = 0.

0: Regular conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the ADC_DR register.

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase. It is cleared by software writing 1 or when ADEN = 0.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it or when ADEN = 0.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

26.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDYIE	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDYIE**: ADC internal voltage regulator interrupt enable

This bit is set and cleared by software to enable/disable the ADC internal voltage regulator interrupt.

0: Internal voltage regulator interrupt disabled

1: Internal voltage regulator interrupt enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the injected context queue overflow interrupt.

0: Injected context queue overflow interrupt disabled

1: Injected context queue overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

26.6.3 ADC control register (ADC_CR)

Address offset: 0x008

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential input mode for the calibration.

0: Single-ended input calibration mode when ADCAL = 1

1: Differential input calibration mode when ADCAL = 1

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0 and ADEN =0).

Bit 28 **ADVREGEN**: ADC internal voltage regulator enable

This bits is set by software to enable the ADC internal voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC internal voltage regulator disabled

1: ADC internal voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 26.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL = 0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be reconfigured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

When a software trigger is used, JADSTART bit is cleared by hardware, but JADSTP bit must be programmed to reconfigure the ADC.

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP command).

It is cleared by hardware when the conversion is effectively discarded. and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

When a software trigger is used, ADSTART bit is cleared by hardware. However it is necessary to write ADSTP bit to reconfigure the ADC.

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop ongoing regular conversions. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use ADSTP).

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (JEXTSEL = 0x0 and JDISEN = 1), this bit is cleared by hardware immediately after injected conversion starts.
- In all cases, after the execution of the JADSTP command, this bit and JADSTP are cleared by hardware simultaneously.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1, ADDIS = 0 and JAUTO = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared).

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (EXTSEL = 0x0 and DISEN = 1), this bit is cleared by hardware immediately after regular conversion starts.
- In all cases, after the execution of the ADSTP command, this bit and ADSTP are cleared by hardware simultaneously.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and place it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN = 1, ADDIS = 0 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all the bits of ADC_CR registers are cleared (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0), except for ADVREGEN bit which must be set. In addition, the software must wait for the startup time of the voltage regulator.

26.6.4 ADC configuration register (ADC_CFGR1)

Address offset: 0x00C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **JQDIS**: Injected queue disable

This bit is set and cleared by software to disable the injected queue mechanism:

0: Injected queue enabled

1: Injected queue disabled

Note: The software is allowed to write this bit only when ADEN = 0, ADSTART = 0, and JADSTART = 0.

Note: A set or reset of JQDIS bit causes the injected queue to be flushed and the ADC_JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10011: ADC analog input channel 19 monitored by AWD1

Others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: ADC_JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: ADC_JSQR mode 0: The Queue is never empty and maintains the last written configuration to ADC_JSQR.

1: ADC_JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 26.4.23: Queue of context for injected conversions* for more information.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: When dual mode is enabled (DUAL bits of ADCC_CCR register are not equal to zero), the JDISCEN bit of the slave ADC is not valid and controlled by the JDISCEN bit of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCNUM[2:0] bits of the slave ADC are not valid and controlled by the DISCNUM[2:0] bits of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When regular dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCEN bit of the slave ADC is not valid and controlled by the DISCEN bit of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the autodelayed conversion mode.

0: Autodelayed conversion mode off

1: Autodelayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the AUTDLY bit of the slave ADC is not valid and controlled by the AUTDLY bit of the master ADC.

Bit 13 **CONT**: Single/continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When regular dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the CONT bit of the slave ADC is not valid and controlled by the CONT bit of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0
00001: adc_ext_trg1
00010: adc_ext_trg2
00011: adc_ext_trg3
00100: adc_ext_trg4
00101: adc_ext_trg5
00110: adc_ext_trg6
00111: adc_ext_trg7
...
11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data management configuration

These bits are set and cleared by software to select how the ADC interface output data are managed.

00: Regular conversion data stored in DR only
01: DMA one-shot mode selected
10: Reserved
11: DMA circular mode selected

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

26.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSHIFT[3:0]				Res.	Res.	OVSR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPTRIG	SWTRIG	BULB	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				Res.	PDIFFREF	PDIFF	Res.	ROVSE
rw	rw	rw			rw	rw	rw	rw	rw	rw		rw	rw		rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **OVSR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled
1: Sampling time control trigger mode enabled

If EXTEN[1:0] bits are set to 01, the sampling time starts on the trigger rising edge, and the conversion starts on the trigger falling edge.

The SMPTRIG bit must not be set when the BULB bit is set.

When EXTEN[1:0] bits are set to 00, set SWTRIG to start the sampling.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with sampling time control trigger mode*.*

Bit 14 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to trigger the conversion in sampling time control trigger mode.

0: Software trigger starts the conversion for sampling time control trigger mode
1: Software trigger starts the sampling for *sampling time control trigger mode*

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing), SMPTRIG = 1, and EXTEN[0:1] = 00.

Bit 13 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

BULB bit must not be set when the SWTRIG bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with bulb mode.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing). It is recommended to clear both JOVSE and GCOMP when ROVSM = 0.

Bit 9 **TROVS**: Triggered regular oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift

0001: 1-bit shift

0010: 2-bit shift

0011: 3-bit shift

0100: 4-bit shift

0101: 5-bit shift

0110: 6-bit shift

0111: 7-bit shift

1000: 8-bit shift

1001: 9-bit shift

1010: 10-bit shift

Other: reserved, must not be used

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bit 3 **PDIFFREF**: Pseudodifferential mode reference voltage selection

This bit is set and cleared by software to select the reference voltage used in pseudodifferential mode.

0: V_{REF-}

1: $V_{REF+}/2$

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 2 **PDIFF**: Pseudodifferential mode enable

This bit is set and cleared by software to enable/disable the pseudodifferential mode.

0: Standard differential mode selected

1: Pseudodifferential mode selected

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ROVSE**: Regular oversampling enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular oversampling disabled

1: Regular oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)

26.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles

001: 4.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 24.5 ADC clock cycles

101: 47.5 ADC clock cycles

110: 256.5 ADC clock cycles

111: 848.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

26.6.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 19 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 4.5 ADC clock cycles
010: 7.5 ADC clock cycles
011: 12.5 ADC clock cycles
100: 24.5 ADC clock cycles
101: 47.5 ADC clock cycles
110: 256.5 ADC clock cycles
111: 848.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

26.6.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **PCSEL[19:0]**: Channel i ($V_{INP[i]}$) preselection

These bits are written by software to preselect the input channel I/O to be converted.

0: Input channel i ($V_{INP[i]}$) is not preselected for conversion, the result of the ADC conversion for this channel is wrong

1: Input channel i ($V_{INP[i]}$) is preselected for conversion

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Configuring the PCSEL bit is not necessary for the internal channels (such as VREFINT).

26.6.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	LEN[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 1st in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **LEN[3:0]:** Regular channel sequence length
These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 9th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 8th in the regular conversion sequence
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 7th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 12th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 11th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 10th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.12 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 16th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 15th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.13 ADC regular data register (ADC_DR)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA[31:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 26.4.28: Data management*.

26.6.14 ADC injected sequence register (ADC_JSQR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bits 6:2 **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JLEN[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JQDIS = 0 or when JADSTART = 0 and JQDIS = 1 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.15 ADC offset y configuration register (ADC_OFCFGRy)

Address offset: 0x050 + 0x004 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET_CH[4:0]					SSAT	USAT	POS OFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:27 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed in bits OFFSET[21:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the data offset y.

Bit 26 **SSAT**: Signed saturation enable

This bit is set and cleared by software to enable the signed saturation feature.(see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*).

0: Offset is subtracted maintaining the data integrity and extending converted data size (13-bit signed format)

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **USAT**: Unsigned saturation enable

This bit is set and cleared by software to enable the unsigned saturation feature.

0: Offset is subtracted maintaining the data integrity

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **POSOFF**: Positive offset enable

This bit is set and cleared by software to enable the positive offset

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:0 Reserved, must be kept at reset value.

26.6.16 ADC offset y register (ADC_OFRy)

Address offset: 0x060 + 0x004 * (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:16]					
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **OFFSET[21:0]:** Data offset y for the channel programmed in OFFSETy_CH[4:0] bits

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the OFFSETy_CH[4:0] bits. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

26.6.17 ADC gain compensation register (ADC_GCOMP)

Address offset: 0x070

Reset value: 0x0000 1000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G COMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GCOMPCOEFF[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GCOMP:** Gain compensation mode

This bit is set and cleared by software to enable the Gain compensation mode.

0: Regular ADC operation mode

1: Gain compensation enabled and applied on all channels.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing)

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:0 **GCOMPCOEFF[13:0]:** Gain compensation coefficient

These bits are set and cleared by software to program the gain compensation coefficient.

00 1000 0000 0000: gain factor of 0.5

...

01 0000 0000 0000: gain factor of 1

10 0000 0000 0000: gain factor of 2

11 0000 0000 0000: gain factor of 3

...

The coefficient is divided by 4096 to get the gain factor ranging from 0 to 3.9999756

Note: This gain compensation is only applied when GCOMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing).

26.6.18 ADC analog watchdog y source channel register (ADC_AWDySR)

Address offset: 0x070 + 0x004 * (y -1), (y = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDS_CH[19:16]			
												rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWDS_CH[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDS_CH[19:0]:** Analog watchdog source for channel *i* (V_{INP}[i])

These bits are set by hardware when channel *i* has generated the flag. They are cleared by software by writing 1 to it, or by setting the ADDIS bit. Each bit corresponds to the input channel that created the analog watchdog flag.

26.6.19 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x080 + 0x004 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **JDATA[31:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left- or right-aligned as described in *Section 26.4.28: Data management*.

26.6.20 ADC configuration register 3 (ADC_CFGR3)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JLSHIFT[3:0]				Res.	Res.	JOVSR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TJOVS	JOVSS[3:0]				Res.	Res.	JOVSIVE	JOVSE	JOVSPAR
						rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:28 **JLSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **JOVSR[9:0]**: Injected channel oversampling ratio

This bitfield is set and cleared by software to define the injected channel oversampling ratio.

00 0000 0000: 1x (no oversampling)

00 0000 0001-11 1111 1111: {val+1}x

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **TJOVS**: Triggered injected channel oversampling

This bit is set and cleared by software to enable triggered injected channel oversampling.

0: All oversampled conversions for a channel are done consecutively following a trigger
1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **JOVSS[3:0]**: Injected channel oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift
0001: Shift 1-bit
0010: Shift 2-bits
0011: Shift 3-bits
0100: Shift 4-bits
0101: Shift 5-bits
0110: Shift 6-bits
0111: Shift 7-bits
1000: Shift 8-bits
Others: reserved

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **JOVSIVE**: Injected oversampling intermediate value enable

This bit is set and cleared by software to enable the access to the injected oversampling intermediate value on JDATA register.

0: Injected oversampling intermediate data not available on JDATA
1: Injected oversampling intermediate data available on JDATA

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 1 **JOVSE**: Injected oversampling enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected oversampling disabled
1: Injected oversampling enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 0 **JOVSPAR**: Parallel injected oversampling enable

This bit is set and cleared by software to enable parallel injected oversampling.

0: Parallel Injected oversampling disabled
1: Parallel Injected oversampling enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing), it is necessary to set JOVSE bit when JOVSPAR bit set.

26.6.21 ADC injected oversampling counter register 1 (ADC_JOVSCR1)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	JOVSC2[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	JOVSC1[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **JOVSC2[9:0]:** Oversampling counter number for injected channel 2
This bitfield is set by hardware and cleared by hardware when JADSTART or ADDIS is set.
00 0000 0000: No oversampling or oversampling conversion finished
00 0000 0001 - 11 1111 1111: Number of remaining conversions to complete the oversampling

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **JOVSC1[9:0]:** Oversampling counter number for injected channel 1
This bitfield is set by hardware and cleared by hardware when JADSTART or ADDIS is set.
00 0000 0000: No oversampling or oversampling conversion finished
00 0000 0001 - 11 1111 1111: Number of remaining conversions to complete the oversampling

26.6.22 ADC injected oversampling counter register 2 (ADC_JOVSCR2)

Address offset: 0x098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	JOVSC4[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	JOVSC3[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **JOVSC4[9:0]:** Oversampling counter number for injected channel 4
This bitfield is set by hardware and cleared by hardware when JADSTART or ADDIS is set.
00 0000 0000: Number of remaining conversions to complete the oversampling
00 0000 0001 - 11 1111 1111: Number of remaining conversions to complete the oversampling

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **JOVSC3[9:0]:** Oversampling counter number for injected channel 3
This bitfield is set by hardware and cleared by hardware when JADSTART or ADDIS is set.
00 0000 0000: No oversampling or oversampling conversion finished
00 0000 0001 - 11 1111 1111: Number of remaining conversions to complete the oversampling

26.6.23 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWDCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDCH[19:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWDCH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

26.6.24 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWDCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDCH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWDCH[19:0] = 000..0, the analog Watchdog 3 is disabled.

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

26.6.25 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR)

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 1 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 1.
Refer to *Section 26.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT).*

26.6.26 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR)

Address offset: 0x0AC

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWDFILT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.
000: No filtering, one detection generates an AWD1 flag or an interrupt
001: two consecutive detections generate an AWD1 flag or an interrupt
...
111: Eight consecutive detections generate an AWD1 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 1 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 1.
Refer to *Analog windows watchdog* section.

26.6.27 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR)

Address offset: 0x0B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 2 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

26.6.28 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR)

Address offset: 0x0B4

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWDFILT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.

000: No filtering, one detection generates an AWD2 flag or an interrupt

001: two consecutive detections generate an AWD2 flag or an interrupt

...

111: Eight consecutive detections generate an AWD2 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 2 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

26.6.29 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR)

Address offset: 0x0B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 3 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

26.6.30 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR)

Address offset: 0x0BC

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWDFILT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.

000: No filtering, one detection generates an AWD3 flag or an interrupt

001: two consecutive detections generate an AWD3 flag or an interrupt

...

111: Eight consecutive detections generate an AWD3 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 3 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

26.6.31 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0x0C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIFSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **DIFSEL[19:0]:** Differential mode for channels 19 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or differential or pseudodifferential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single-ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in differential or pseudodifferential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

26.6.32 ADC calibration factors (ADC_CALFACT)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **CALFACT_D[6:0]**: Calibration factors in differential mode

These bits are written by hardware or by software.

Once a differential input calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored in the analog ADC, it is applied once a new differential conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no calibration is ongoing and no conversion is ongoing).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT_S[6:0]**: Calibration factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended input calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored in the analog ADC, it is applied once a new single-ended conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no calibration is ongoing and no conversion is ongoing).

This bit can only be cleared by a power-on reset.

26.6.33 ADC option register (ADC_OR)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP7	OP6	OP5	OP4	OP3	OP2	OP1	OP0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **OP7:** Option bit 7

This bit is set and cleared by software to select the optional trigger source
Condition: ADC1 and ADC2
0: Reserved
1: Reserved
Condition: ADC3
0: adc_jext_trg9 = tim8_oc4
1: adc_jext_trg9 = tim4_trgo

Note: The software is allowed to write this bit only when the JADSTART = 0.

Bit 6 **OP6:** Option bit 6

This bit is set and cleared by software to select the optional trigger source
Condition: ADC1 and ADC2
0: adc_jext_trg9 = tim3_trgo
1: adc_jext_trg9 = tim4_trgo. This bit affects both ADC1 and ADC2.
Condition: ADC3
0: adc_jext_trg8 = tim8_oc2
1: adc_jext_trg8 = tim4_oc3

Note: The software is allowed to write this bit only when the JADSTART = 0.

Bit 5 **OP5:** Option bit 5

This bit is set and cleared by software to select the optional trigger source
0: adc_ext_trg9 = tim3_trgo
1: adc_ext_trg9 = tim4_trgo. This bit affects both ADC1 and ADC2; ADC3 has independent control.

Note: The software is allowed to write this bit only when the ADSTART = 0.

Bit 4 **OP4:** Option bit 4

This bit is set and cleared by software to select the optional trigger source
Condition: ADC1 and ADC2
0: adc_ext_trg8 = tim3_oc4
1: adc_ext_trg8 = tim4_oc4. This bit affects both ADC1 and ADC2
Condition: ADC3
0: adc_ext_trg8 = tim3_oc2
1: adc_ext_trg8 = tim4_oc1

Note: The software is allowed to write this bit only when the ADSTART = 0.

Bit 3 **OP3:** Option bit 3

This bit is set and cleared by software to enable/disable the internal reference buffer.

0: Internal reference buffer disabled

1: Internal reference buffer enabled

Note: The software is allowed to write this bit only when the ADC is disabled.

Bit 2 **OP2:** Option bit 2

This bit is set and cleared by software to select the redundancy bit.

0: Redundancy on bit 5

1: Redundancy on bit 3

Note: The software is allowed to write this bit only when the ADC is disabled.

Bit 1 **OP1:** Option bit 1

This bit is set and cleared by software to enable/disable the redundancy.

0: Redundancy enabled

1: Redundancy disabled

Note: The software is allowed to write this bit only when the ADC is disabled.

Bit 0 **OP0:** Option bit 0

This bit is set and cleared by software to enable/disable the V_{DDCORE} channel.

0: V_{DDCORE} disabled

1: V_{DDCORE} enabled

26.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs.

26.7.1 ADC common status register (ADCC_CSR)

Address offset: 0x000

Reset value: 0x0000 0000

This register provides an image of the flags of the different ADCs. Nevertheless it is read-only and does not allow to clear the different flags. Instead each flag must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	LDORDY_SLV	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
			r		r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDY_MST	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
			r		r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **LDORDY_SLV:** ADC internal voltage regulator flag of the slave ADC

This bit is a copy of the LDORDY bit in the corresponding ADC_ISR register.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **JQOVF_SLV:** Injected context queue overflow flag of the slave ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17 **EOSMP_SLV:** End of sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY_MST:** ADC internal voltage regulator flag of the master ADC
This bit is a copy of the LDORDY bit in the corresponding ADC_ISR register.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVF_MST:** Injected context queue overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

26.7.2 ADC common control register (ADCC_CCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSEN	VREFEN	Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw			rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TSEN**: Temperature sensor voltage enable
This bit is set and cleared by software to control V_{SENSE} channel.
0: Temperature sensor channel disabled
1: Temperature sensor channel enabled
Note: The temperature sensor is not available on all ADC instances. Refer to Section 26.4.4: ADC connectivity for details.
Note: The software is allowed to write this bit only when ADEN = 0.

Bit 22 **VREFEN**: V_{REFINT} enable
This bit is set and cleared by software to enable/disable the V_{REFINT} channel.
0: V_{REFINT} channel disabled
1: V_{REFINT} channel enabled
Note: V_{REFINT} is not available on all ADC instances. Refer to Section 26.4.4: ADC connectivity for details.
Note: The software is allowed to write this bit only when ADEN = 0.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:14 **DAMDF[1:0]:** Dual ADC mode data format

This bitfield are set and cleared by software. It specifies the data format in the common data register ADCC_CDR and ADCC_CDR2.

00: Dual ADC mode without data packing (ADCC_CDR and ADCC_CDR2 registers not used).

01: Reserved

10: Data formatting mode for any data width (ADCC_CDR data register is used when the data width is less than 16 bits, otherwise ADCC_CDR2 register is used)

11: Data formatting mode for data width lower that 8 bits (ADCC_CDR data register is used)

Note: The software is allowed to write these bits only when ADEN = 0 (ADC is disabled).

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY[3:0]:** Delay between two sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 238* for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.

All the ADCs independent:

00000: Independent mode

The following settings apply to dual mode, master and slave ADCs working together

00001: Combined regular simultaneous + Injected simultaneous mode

00010: Combined regular simultaneous + Alternate trigger mode

00011: Combined interleaved mode + Injected simultaneous mode

00101: Injected simultaneous mode only

00110: Regular simultaneous mode only

00111: Interleaved mode only

01001: Alternate trigger mode only

Others: Reserved

All other combinations are reserved and must not be programmed

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

26.7.3 ADC common regular data register for dual mode (ADCC_CDR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_SLV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_MST[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 26.4.33: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 26.4.33: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

26.7.4 ADC common regular data register for dual mode (ADCC_CDR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_ALT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_ALT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA_ALT[31:0]**: Regular data of the master/slave alternated ADCs.

In dual mode, these bits contain the regular 32-bit data of the master and the slave ADC. Refer to *Section 26.4.33: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

26.8 ADC register map

ADC1 and ADC3 are master ADCs.

ADC2 operates as a slave from ADC1.

Table 240. ADC register map and reset values for each ADC

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDY	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																				0		0	0	0	0	0	0	0	0	0	0	0

Table 240. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDYIE	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																				0		0	0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR	ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0																							0	0	0	0	0	0
0x00C	ADC_CFGR1	JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	Reset value	0x8000 0000																															
0x010	ADC_CFGR2	LSHIFT[3:0]				Res.	Res.	OVSR[9:0]										SMPTRIG	SWTRIG	BULB	Res.	Res.	ROMVSM	TROVS	OVSS[3:0]				Res.	PDIFFREF	PDIFF	Res.	ROVSE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0		0	0		0
0x014	ADC_SMPR1	Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ADC_SMPR2	Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:0]			SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ADC_PCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x02C	Reserved																																
0x030	ADC_SQR1	Res.	Res.	Res.	SQ4[4:0]				Res.	SQ3[4:0]				Res.	SQ2[4:0]				Res.	SQ1[4:0]				Res.	Res.	LEN[3:0]							
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x034	ADC_SQR2	Res.	Res.	Res.	SQ9[4:0]				Res.	SQ8[4:0]				Res.	SQ7[4:0]				Res.	SQ6[4:0]				Res.	Res.	SQ5[4:0]							
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0	0
0x038	ADC_SQR3	Res.	Res.	Res.	SQ14[4:0]				Res.	SQ13[4:0]				Res.	SQ12[4:0]				Res.	SQ11[4:0]				Res.	Res.	SQ10[4:0]							
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0	0
0x03C	ADC_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x040	ADC_DR	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044-0x48	Reserved																																
0x04C	ADC_JSQR	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]				Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 240. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	ADC_OFCFGR1			OFFSET1_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x054	ADC_OFCFGR2			OFFSET2_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x058	ADC_OFCFGR3			OFFSET3_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x05C	ADC_OFCFGR4			OFFSET4_CH[4:0]			SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x060	ADC_OFR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	ADC_OFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	ADC_OFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	ADC_OFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	ADC_GCOMP	GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMPCOEFF[13:0]													
	Reset value	0																		0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x074	ADC_AWD1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDS_CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	ADC_AWD2SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDS_CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	ADC_AWD3SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDS_CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	ADC_JDR1	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	ADC_JDR2	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 240. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	ADC_JDR3	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	ADC_JDR4	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	ADC_CFGR3	JLSHIFT[3:0]				Res.	Res.	JOVSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	TJOVS	JOVSS[3:0]				Res.	Res.	JOVSIVE	JOVSE	JOVSPAR
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0							0	0	0	0	0			0	0	0
0x094	ADC_JOVSCR1	Res.	Res.	Res.	Res.	Res.	Res.	JOVSC2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	JOVSC1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x098	ADC_JOVSCR2	Res.	Res.	Res.	Res.	Res.	Res.	JOVSC4[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	JOVSC3[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	ADC_AWD1LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	ADC_AWD1HTR	AWDFILT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B0	ADC_AWD2LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	ADC_AWD2HTR	AWDFIL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B8	ADC_AWD3LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC	ADC_AWD3HTR	AWDFIL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C0	ADC_DIFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8-0x0CC	Reserved																																
0x0D0	ADC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP7	OP6	OP5	OP4	OP3	OP2	OP1	OP0
	Reset value																									0	0	0	0	0	0	0	0
0x0D4-0x0FC	Reserved																																

Table 241. ADC register map and reset values (master and slave ADC common registers)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADCC_CSR	Res.	Res.	Res.	LDORDY_SLV	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	LDORDY_MST	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					slave ADC																master ADC												
	Reset value				0		0	0	0	0	0	0	0	0	0	0	0				0		0	0	0	0	0	0	0	0	0	0	0
0x004	Reserved	Res.																															
0x008	ADCC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	TSEN	VREFEN	Res.				Res.	Res.	DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value									0	0							0	0			0	0	0	0				0	0	0	0	0
0x00C	ADCC_CDR	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	ADCC_CDR2	RDATA_ALT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

27 Digital temperature sensor (DTS)

27.1 DTS introduction

The device embeds a sensor that converts the temperature into a square wave which frequency is proportional to the temperature. The frequency is measured either with the PCLK or the LSE clock.

27.2 DTS main features

The temperature sensor block main features are the following:

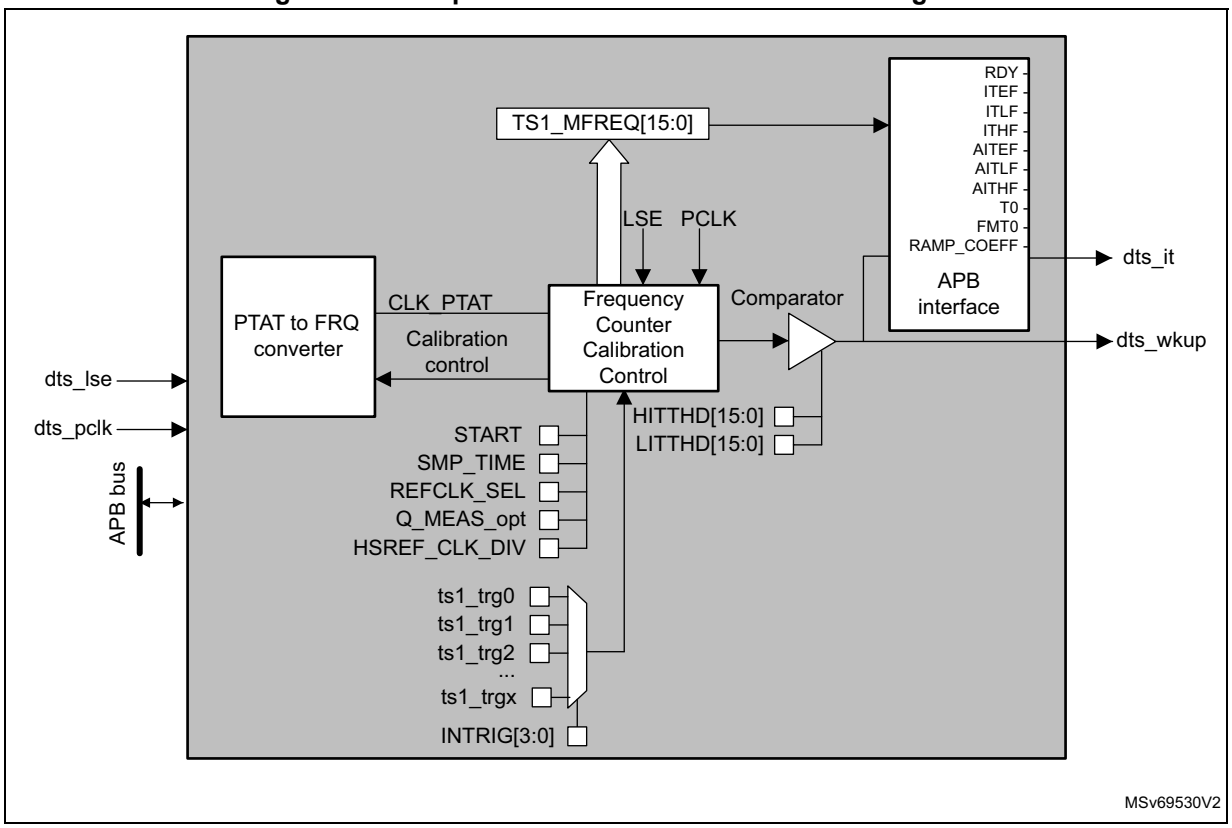
- Start of measurement triggered by software or 4 hardware sources
- Programmable sampling time to increase temperature measurement precision
- Counter synchronized on LSE or PCLK clock
- Temperature watchdog on low and high threshold
- Interrupt generation when the temperature is lower or higher than predefined thresholds and at the end of measurement.
- Asynchronous wake-up signal generation when the temperature is higher/lower than a predefined threshold (LSE mode only)
- Quick measurement using LSE clock

27.3 DTS functional description

27.3.1 DTS block diagram

The temperature sensor block diagram is shown in *Figure 262*.

Figure 262. Temperature sensor functional block diagram



27.3.2 DTS internal signals

Table 242. DTS internal input/output signals

Signal name	Signal type	Description
dts_lse	Digital input	LSE clock
dts_pclk	Digital input	APB clock
dts_it	Digital output	Temperature sensor interrupt
dts_wkup	Digital output	Temperature sensor wake-up

27.3.3 DTS block operation

The analog part of the temperature sensor outputs a frequency that is proportional to the absolute temperature (CLK_PTAT). The frequency measurement is based on the PCLK or the LSE clock.

Before each measurement, the temperature sensor performs a calibration of the frequency generation blocks.

27.3.4 Operating modes

Several operating modes can be selected by setting the REFCLK_SEL bit in *Temperature sensor configuration register 1 (DTS_CFGR1)*:

- PCLK only (REFCLK_SEL = 0)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using PCLK clock
- PCLK and LSE (REFCLK_SEL = 1)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using the LSE clock.
- LSE only (REFCLK_SEL = 1) and PCLK OFF

  The registers cannot be accessed. The measurement can be performed using the LSE clock. This mode is used to exit from Sleep mode by using hardware triggers and the asynchronous interrupt line.

27.3.5 Calibration

The temperature sensor must run the calibration prior to any frequency measurement. The calibration is performed automatically when the temperature measurement is triggered except for quick measurement mode (Q_MEAS_OPT set to 1 in DTS_CFGR1).

27.3.6 Prescaler

When a calibration is ongoing, the counter clock must be slower than 1 MHz. This is achieved by the PCLK clock prescaler embedded in the temperature sensor.

During the temperature measurement period, the prescaler is bypassed.

- When PCLK is used as reference clock (REFCLK_SEL set to 0 in DTS_CFGR1), a prescaler is used. Its division ratio must be configured up to 127 (refer to the HSREF_CLK_DIV[6:0] register definition for the divider setting).
- When LSE is used as reference clock (REFCLK_SEL set to 1 in DTS_CFGR1), the timebase is equal to 2 LSE periods. In this case, no prescaler is used.

27.3.7 Temperature measurement principles

The analog part of temperature sensor outputs a signal (CLK_PTAT) which FM(T) frequency is temperature-dependent.

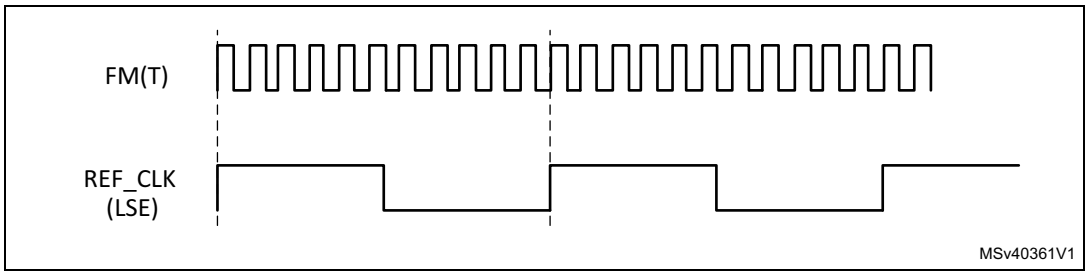
Either PCLK or LSE can be selected as reference clock (REF_CLK) through the REFCLK_SEL bit in DTS_CFGR1.

The counting method depends on the REF_CLK frequency. This is due to the fact that two counters are implemented in the temperature sensor block:

- For low REF_CLK frequencies, a counting of FM(T) cycles is performed during one or several REF_CLK cycles.
- For high REF_CLK frequencies, a counting of REF_CLK cycles is performed during one or several FM(T) cycles.

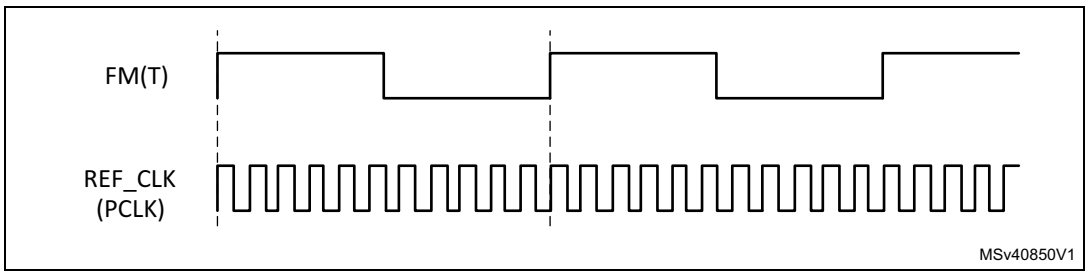
This counter behavior is shown in *Figure 263* and *Figure 264*.

Figure 263. Method for low REF_CLK frequencies



1. To increase the precision, FM(T) measurement can be done on several LSE periods.

Figure 264. Method for high REF_CLK frequencies



1. To increase the precision, PCLK measurement can be done on several FM(T) periods.

The counting result is stored in the DTS_DR register (see *Temperature sensor data register (DTS_DR)*.

Once the FM(T) frequency has been obtained, the corresponding temperature can be calculated by software using the following formula:

- When PCLK is used:

$$T = T_0 + ((F_{PCLK} / TS1_MFREQ) \times TS1_SMP_TIME - 100 \times TS1_FMT0) / TS1_RAMP_COEFF$$

where

T0 (factory calibration temperature) is equal to 30 °C.

TS1_FMT0 is measured and stored in the DTS_T0VALR1 register. It is expressed in hundreds of Hertz.

TS1_RAMP_COEFF is measured during tests in factory and stored in DTS_RAMPVALR register. This value is expressed in Hz/°C.

- When the LSE clock is used

$$T = T_0 + ((F_{LSE} \times TS1_MFREQ / TS1_SMP_TIME) - (100 \times TS1_FMT0)) / TS1_RAMP_COEFF$$

27.3.8 Sampling time

The sampling period can be increased to improve measurement accuracy. This is useful when the reference frequency (REF_CLK) is close to the FM(T) frequency. The default value is one REF_CLK cycle in LSE mode, and one FM(T) cycle in PCLK mode.

The sampling time is configured through TS1_SMP_TIME bits in DTS_CFGR1 register (see *Table 243*).

Table 243. Sampling time configuration

TS1_SMP_TIME[3:0]	LSE or FM(T) clock cycle(s)
0000	1
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

27.3.9 Quick measurement mode

If a high precision is not required, the calibration step included in each measurement sequence can be skipped by setting Q_MEAS_OPT to 1 in the DTS_CFGR1 register. This method must used only when the LSE clock is selected as reference clock (LSREF_CLK set to 1). This mode can reduce the measurement time.

27.3.10 Trigger input

The temperature measurement can be triggered either by software or by an external event. The trigger source can be selected through TS1_INTRIG[3:0] bits in DTS_CFGR1.

Table 244. Trigger configuration

Name	TS1_INTRIG[3:0]				Comment
N.A	0	0	0	0	No hardware trigger
ts1_trg0	0	0	0	1	lptim1_ch1
ts1_trg1	0	0	1	0	Reserved
ts1_trg2	0	0	1	1	tim26_trg_ck
ts1_trg3	0	1	0	0	exti13
ts1_trg4	0	1	0	1	Reserved
ts1_trg5	0	1	1	0	
ts1_trg6	0	1	1	1	
ts1_trg7	1	0	0	0	
ts1_trg8	1	0	0	1	
ts1_trg9	1	0	1	0	
ts1_trg10	1	0	1	1	
ts1_trg11	1	1	0	0	
ts1_trg12	1	1	0	1	
ts1_trg13	1	1	1	0	
ts1_trg14	1	1	1	1	

Note: Hardware triggers are active only on the rising edge.

The temperature sensor can only capture a hardware trigger rising edge when TS1_RDY bit is set (see Section 27.3.11: On-off control and ready flag), otherwise the trigger is ignored.

If a trigger source changes on-the-fly, the new trigger source signal must be low. If the new source signal is high, the temperature sensor detects a rising edge and start the measurement sequence.

27.3.11 On-off control and ready flag

The DTS block can be enabled by setting TS1_EN bit in DTS_CFGR1 register. The TS1_RDY flag in the *Temperature sensor status register (DTS_SR)* indicate that the DTS block is ready for temperature measurement: when TS1_RDY bit is set to 1, the measurement can be started. Once a measurement has started, TS1_RDY bit is reset. New measurement requests are then ignored. Once the measurement is finished, TS1_RDY bit is set again to indicate the sensor is ready to start a new measurement.

27.3.12 Temperature measurement sequence

Start of measurement can be triggered by software or hardware.

Software trigger

The software trigger is selected when TS1_INTRIG_SEL[3:0] is set to ‘0000’ in DTS_CFGR1.

If TS1_RDY is set to 1, writing TS1_START bit to 1 in DTS_CFGR1 starts the measurement.

If TS1_RDY equals 0, the software trigger does not start until TS1_RDY is set.

If TS1_START bit is kept at 1 once the measurement is finished, then the TS1_RDY flag become 1 and the measurement restarts.

Hardware trigger

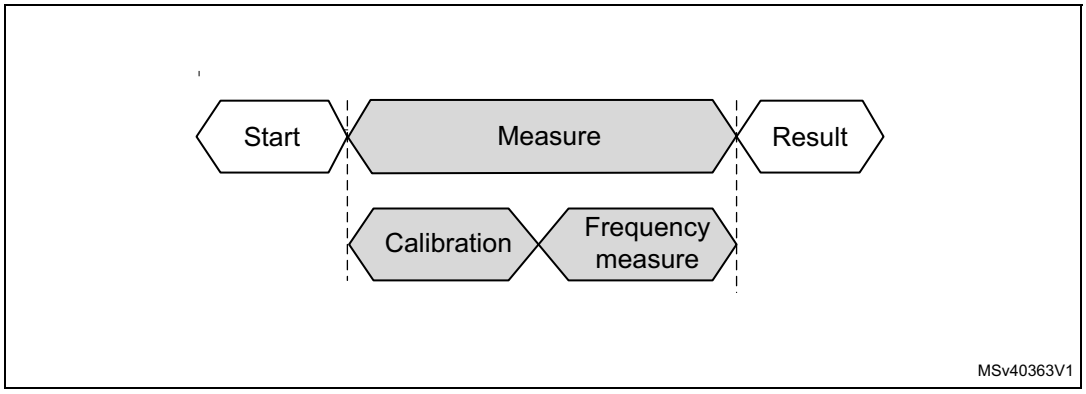
TS1_INTRIG_SEL[3:0] bits allow selecting one hardware trigger out of 4. If TS1_RDY is set to 1, a rising edge on the trigger signal starts the measurement. When TS1_RDY is 0, the rising edge is ignored.

Temperature measurement sequence

One measurement contains two steps: the calibration of the analog blocks and the measurement. The calibration automatically starts when the measurement is triggered (see *Section 27.3.5: Calibration*). The measurement period depends on the following DTS_CFGR1 bits:

- the reference clock selected through REFCLK_SEL bit
- the divider ratio configured by HSREF_CLK_DIV bits
- the sampling time defined by TS1_SMP_TIME bits.

Figure 265. Temperature sensor sequence



27.4 DTS low-power modes

Table 245. Temperature sensor behavior in low-power modes

Mode	Description
Sleep	Only works in LSE mode. DTS interrupt causes the device to exit from Sleep mode.
Stop	Only works in LSE mode. DTS interrupt cause the device to exit from Stop mode.

27.5 DTS interrupts

There are two ways to use the DTS block as an interrupt source.The DTS interrupt line can be connected to the EXTI controller (see *Section 27.5.3: Asynchronous wake-up*) or to the CPU NVIC (see *Section 27.5.2: Synchronous interrupt*).

27.5.1 Temperature window comparator

The DTS_ITR1 register allows defining the high and low threshold that are used for temperature comparison. If the temperature data is equal or higher than TS1_HITTHD, or equal or lower than TS1_LITTHD bit, an interrupt is generated and the corresponding flag, TS1_ ITLF, TS1_ITHF, TS1_AITLF and TS1_AITHF, is set in the DTS_SR register (see *Section 27.6.6*).

27.5.2 Synchronous interrupt

A global interrupt output line is available on the DTS block. The interrupt can be generated at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 27.5.1: Temperature window comparator*).

Three interrupt events can be select via 3 bits in DTS_ITENR register (see *Section 27.6.7*). All combinations of interrupts are allowed.

The TS1_ITEF, TS1_ITLF and TS1_ITHF flags in the DTS_SR register reflect the interrupt event. They can be reset with the correspond bits of the DTS_ICIFR register (see *Section 27.6.8*).

27.5.3 Asynchronous wake-up

The DTS block also provides an asynchronous interrupt line. It is used only when the LSE is selected as reference clock (REFCLK_SEL=1).

This line can generate a signal that wakes up the system from Sleep mode at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 27.5.1: Temperature window comparator*).

Three asynchronous wake-up events can be selected via 3 bits in DTS_ITENR register. All combination of interrupts are allowed.

The TS1_AITEF, TS1_AITLF and TS1_AITHF flags in the DTS_SR register reflect the interrupt status. They can be reset with the correspond bits of the DTS_ICIFR register.

The following table shows the interrupt bits an the their description.

27.6 DTS registers

The registers of this peripheral can only be accessed by-word (32-bit).

27.6.1 Temperature sensor configuration register 1 (DTS_CFGR1)

DTS_CFGR1 is the configuration register for temperature sensor 1.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HSREF_CLK_DIV[6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_SMP_TIME[3:0]			
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TS1_INTRIG_SEL[3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
				rw	rw	rw	rw				rw				rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSREF_CLK_DIV[6:0]**: High speed clock division ratio

These bits are set and cleared by software. They can be used to define the division ratio for the main clock in order to obtain the internal frequency lower than 1 MHz required for the calibration. They are applicable only for calibration when PCLK is selected as reference clock (REFCLK_SEL=0).

0000000: No divider

0000001: No divider

0000010: 1/2 division ratio

...

1111111: 1/127 division ratio

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **Q_MEAS_OPT**: Quick measurement option bit

This bit is set and cleared by software. It is used to increase the measurement speed by suppressing the calibration step. It is effective only when the LSE clock is used as reference clock (REFCLK_SEL=1).

0: Measurement with calibration

1: Measurement without calibration

Bit 20 **REFCLK_SEL**: Reference clock selection bit

This bit is set and cleared by software. It indicates whether the reference clock is the high speed clock (PCLK) or the low speed clock (LSE).

0: High speed reference clock (PCLK)

1: Low speed reference clock (LSE)

Bits 19:16 **TS1_SMP_TIME[3:0]**: Sampling time for temperature sensor 1
These bits allow increasing the sampling time to improve measurement precision.
When the PCLK clock is selected as reference clock (REFCLK_SEL = 0), the measurement is performed at TS1_SMP_TIME period of CLK_PTAT.
When the LSE is selected as reference clock (REFCLK_SEL = 1), the measurement is performed at TS1_SMP_TIME period of LSE.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TS1_INTRIG_SEL[3:0]**: Input trigger selection bit for temperature sensor 1
These bits are set and cleared by software. They select which input triggers a temperature measurement. Refer to *Section 27.3.10: Trigger input*.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TS1_START:** Start frequency measurement on temperature sensor 1
This bit is set and cleared by software.
0: No software trigger.
1: Software trigger for a frequency measurement. (only if TS1 is ready).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TS1_EN:** Temperature sensor 1 enable bit
This bit is set and cleared by software.
0: Temperature sensor 1 disabled
1: Temperature sensor 1 enabled

Note: Once enabled, the temperature sensor is active after a specific delay time. The TS1_RDY flag is set when the sensor is ready.

27.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)

DTS_T0VALR1 contains the value of the factory calibration temperature (T0) for temperature sensor 1. The reset value is factory trimmed.

Address offset: 0x08

Reset value: 0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_FMT0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **TS1_T0[1:0]**: Engineering value of the T0 temperature for temperature sensor 1.
00: 30 °C
01: 130 °C
Others: Reserved, must not be used.

Bits 15:0 **TS1_FMT0[15:0]**: Engineering value of the frequency measured at T0 for temperature sensor 1
This value is expressed in 0.1 kHz.

27.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)

The DTS_RAMPVALR is the ramp coefficient for the temperature sensor. The reset value is factory trimmed.

Address offset: 0x10

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_RAMP_COEFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_RAMP_COEFF[15:0]**: Engineering value of the ramp coefficient for the temperature sensor 1.

This value is expressed in Hz/°C.

27.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)

DTS_ITR1 contains the threshold values for sensor 1.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS1_HITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_LITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TS1_HITTHD[15:0]:** High interrupt threshold for temperature sensor 1

These bits are set and cleared by software. They indicate the highest value than can be reached before raising an interrupt signal.

Bits 15:0 **TS1_LITTHD[15:0]:** Low interrupt threshold for temperature sensor 1

These bits are set and cleared by software. They indicate the lowest value than can be reached before raising an interrupt signal.

27.6.5 Temperature sensor data register (DTS_DR)

The DTS_DR contains the number of REF_CLK cycles used to compute the FM(T) frequency.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_MFREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_MFREQ[15:0]**: Value of the counter output value for temperature sensor 1

27.6.6 Temperature sensor status register (DTS_SR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_ RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_ AITHF	TS1_ AITLF	TS1_ AITEF	Res.	TS1_ ITHF	TS1_ ITLF	TS1_ ITEF
r									r	r	r		r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TS1_RDY:** Temperature sensor 1 ready flag
This bit is set and reset by hardware.
It indicates that a measurement is ongoing.
0: Temperature sensor 1 busy
1: Temperature sensor 1 ready

Bits 14:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHF:** Asynchronous interrupt flag for high threshold on temperature sensor 1
This bit is set by hardware when the high threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITHFEN bit is set

Bit 5 **TS1_AITLF:** Asynchronous interrupt flag for low threshold on temperature sensor 1
This bit is set by hardware when the low threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITLFEN bit is set

Bit 4 **TS1_AITEF:** Asynchronous interrupt flag for end of measure on temperature sensor 1
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS1_CAITEF bit in the DTS_ICIFR register.
0: End of measure not detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_AITEFEN bit is set

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHF:** Interrupt flag for high threshold on temperature sensor 1, synchronized on PCLK
This bit is set by hardware when the high threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITHFEN bit is set

Bit 1 **TS1_ITLF:** Interrupt flag for low threshold on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when the low threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITLFEN bit is set

Bit 0 **TS1_ITEF:** Interrupt flag for end of measurement on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS2_CITEF bit in the DTS_ICIFR register.
0: No end of measurement detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_ITEFEN bit is set

27.6.7 Temperature sensor interrupt enable register (DTS_ITENR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHEN	TS1_AITLEN	TS1_AITEEN	Res.	TS1_ITHEN	TS1_ITLEN	TS1_ITEEN
									rw	rw	rw		rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHEN:** Asynchronous interrupt enable flag on high threshold for temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is above the high threshold (only when REFCLK_SEL= 1")
0: Asynchronous interrupt on high threshold disabled for temperature sensor 1
1: Asynchronous interrupt on high threshold enabled for temperature sensor 1

Bit 5 **TS1_AITLEN:** Asynchronous interrupt enable flag for low threshold on temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is below the low threshold (only when REFCLK_SEL= 1)
0: Asynchronous interrupt on low threshold disabled for temperature sensor 1
1: Asynchronous interrupt on low threshold enabled for temperature sensor 1

Bit 4 **TS1_AITEEN:** Asynchronous interrupt enable flag for end of measurement on temperature sensor 1
This bit are set and cleared by software.
It enables the asynchronous interrupt for end of measurement (only when REFCLK_SEL = 1).
0: Asynchronous interrupt for end of measurement disabled on temperature sensor 1
1: Asynchronous interrupt for end of measurement enabled on temperature sensor 1

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHEN:** Interrupt enable flag for high threshold on temperature sensor 1, synchronized on PCLK.
This bit are set and cleared by software.
It enables the interrupt when the measure reaches or is above the high threshold.
0: Synchronous interrupt for high threshold disabled on temperature sensor 1
1: Synchronous interrupt for high threshold enabled on temperature sensor 1

Bit 1 **TS1_ITLEN:** Interrupt enable flag for low threshold on temperature sensor 1, synchronized on PCLK.
This bit are set and cleared by software.
It enables the synchronous interrupt when the measure reaches or is below the low threshold.
0: Synchronous interrupt for low threshold disabled on temperature sensor 1
1: Synchronous interrupt for low threshold enabled on temperature sensor 1

Bit 0 **TS1_ITEEN:** Interrupt enable flag for end of measurement on temperature sensor 1, synchronized on PCLK.
This bit are set and cleared by software.
It enables the synchronous interrupt for end of measurement.
0: Synchronous interrupt for end of measurement disabled on temperature sensor 1
1: Synchronous interrupt for end of measurement enabled on temperature sensor 1

27.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)

DTS_ICIFR is the control register for the interrupt flags.

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_ CAITHF	TS1_ CAITLF	TS1_ CAITEF	Res.	TS1_ CITHF	TS1_ CITLF	TS1_ CITEF
									rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_CAITHF:** Asynchronous interrupt clear flag for high threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITHF flag in the DTS_SR register.

Bit 5 **TS1_CAITLF:** Asynchronous interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITLF flag in the DTS_SR register.

Bit 4 **TS1_CAITEF:** Write once bit. Clear the asynchronous IT flag for End Of Measure for thermal sensor 1.
Writing 1 clears the TS1_AITEF flag of the DTS_SR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_CITHF:** Interrupt clear flag for high threshold on temperature sensor 1
Writing this bit to 1 clears the TS1_ITHF flag in the DTS_SR register.

Bit 1 **TS1_CITLF:** Interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_ITLF flag in the DTS_SR register.

Bit 0 **TS1_CITEF:** Interrupt clear flag for end of measurement on temperature sensor 1
Writing 1 to this bit clears the TS1_ITEF flag in the DTS_SR register.

27.6.9 Temperature sensor option register (DTS_OR)

The DTS_OR contains general-purpose option bits.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS_OP 31	TS_OP 30	TS_OP 29	TS_OP 28	TS_OP 27	TS_OP 26	TS_OP 25	TS_OP 24	TS_OP 23	TS_OP 22	TS_OP 21	TS_OP 20	TS_OP 19	TS_OP 18	TS_OP 17	TS_OP 16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS_OP 15	TS_OP 14	TS_OP 13	TS_OP 12	TS_OP 11	TS_OP 10	TS_OP 9	TS_OP 8	TS_OP 7	TS_OP 6	TS_OP 5	TS_OP 4	TS_OP 3	TS_OP 2	TS_OP 1	TS_OP 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TS_OP[31:0]:** general purpose option bits

27.6.10 DTS register map

The following table summarizes the temperature sensor registers.

Table 246. DTS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DTS_ CFGR1	Res.	HSREF_CLK_DIV [6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_ SMP_TIME [3:0]				Res.	Res.	Res.	Res.	TS1_ INTRIG_SEL [3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0					0	0	0	0				0				0
0x04	Reserved																																
0x08	DTS_ T0VALR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]		TS1_FMT0[15:0]															
	Reset value															X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	Reserved																																
0x10	DTS_ RAMPVALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RAMP_COEFF[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	DTS_ITR1	TS1_HITTHD[15:0]																TS1_LITTHD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	Reserved																																
0x1C	DTS_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_MFREQ[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DTS_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHF	TS1_AITLF	TS1_AITEF	Res.	TS1_ITHF	TS1_ITLF	TS1_ITEF
	Reset value																	0									0	0	0		0	0	0
0x24	DTS_ITENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHEN	TS1_AITLEN	TS1_AITEEN	Res.	TS1_ITHEN	TS1_ITLEN	TS1_ITEEN
	Reset value																										0	0	0		0	0	0
0x28	DTS_ICIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_CAITHF	TS1_CAITLF	TS1_CAITEF	Res.	TS1_CITHF	TS1_CITLF	TS1_CITEF
	Reset value																										0	0	0		0	0	0

Table 246. DTS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	DTS_OR	TS_OP31	TS_OP30	TS_OP29	TS_OP28	TS_OP27	TS_OP26	TS_OP25	TS_OP24	TS_OP23	TS_OP22	TS_OP21	TS_OP20	TS_OP19	TS_OP18	TS_OP17	TS_OP16	TS_OP15	TS_OP14	TS_OP13	TS_OP12	TS_OP11	TS_OP10	TS_OP9	TS_OP8	TS_OP7	TS_OP6	TS_OP5	TS_OP4	TS_OP3	TS_OP2	TS_OP1	TS_OP0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

28 Digital-to-analog converter (DAC)

28.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features one single channel. Two input reference pins, VREFDAC+ and VREFDAC-, are available for better resolution.

The DACx_OUT1 pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to an on-chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. A calibration can be applied on the buffered DAC output channel. The DAC output channel supports a low power mode, the sample and hold mode.

28.2 DAC main features

The DAC main features are the following (see *Figure 266: DAC block diagram*):

- One DAC interface, one output channel
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Sawtooth wave generation
- Extended sawtooth wave generation
- DMA capability including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- The DAC output can be disconnected from the DACx_OUT1 output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from VREFDAC+ and VREFDAC- pins

Figure 266 shows the block diagram of a DAC channel and *Table 248* gives the pin description.

28.3 DAC implementation

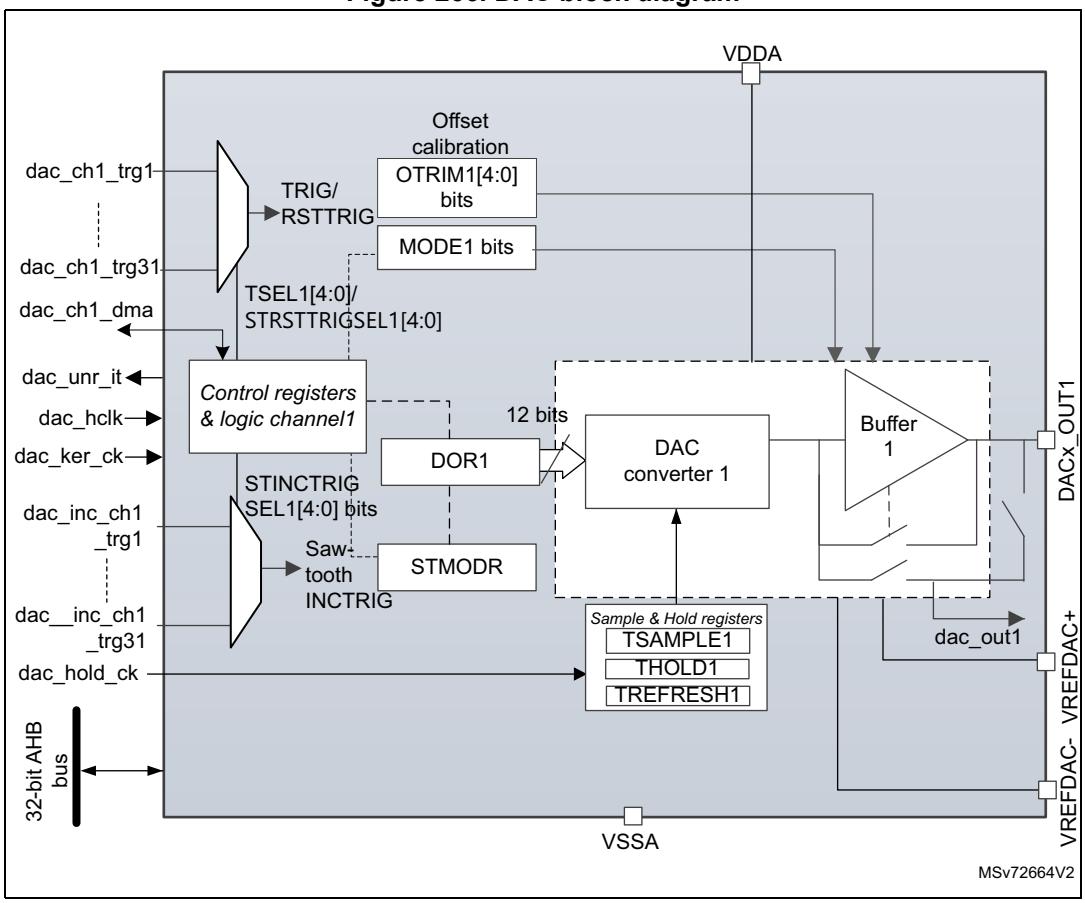
Table 247. DAC features

DAC features	DAC1/2	DAC5/6/7/8/9/10
Dual channel	-	
Output buffer	X	-
I/O connection	DAC1_OUT1 on PA4 DAC2_OUT1 on PA5	-
Maximum sampling time	1 Msps	15 Msps
Autonomous mode	-	
VREFDAC+ pin	X	
Extended sawtooth wave generation	X	
Sample and hold mode	X	-
Option register	-	X

28.4 DAC functional description

28.4.1 DAC block diagram

Figure 266. DAC block diagram



1. MODE1 bits in the DAC_MCR control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.

28.4.2 DAC pins and internal signals

The DAC includes:

- One output channel
- The DACx_OUT1 can be disconnected from the output pin and used as an ordinary GPIO.
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI/LSE clock source (dac_hold_ck) for static conversion.

The DAC includes one output channel. The output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case,

the DAC output channel can be disconnected from the DACx_OUT1 output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The sample and hold block and its associated registers can run in Stop mode using the LSI/LSE clock source (dac_hold_ck).

Table 248. DAC input/output pins

Pin name	Signal type	Remarks
VREFDAC+	Input, analog positive reference	The higher/positive reference voltage for the DAC, $V_{REFDAC+} \leq V_{DDAmax}$ (refer to the product datasheet)
VREFDAC-	Input, analog negative reference	The lower/negative reference voltage for the DAC.
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DACx_OUT1	Analog output signal	DACx channel1 analog output

Table 249. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 31)	Inputs	DAC channel1 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_ker_ck	Input	DAC kernel clock
dac_hold_ck	Input	DAC low-power clock used in sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals

Table 250. DAC interconnection

Signal name	Source	Source type
dac_chx_trg1	tim1_trgo	Internal signal from on-chip timer
dac_chx_trg2	tim6_trgo	Internal signal from on-chip timer
dac_chx_trg3	tim7_trgo	Internal signal from on-chip timer
dac_chx_trg4	tim8_trgo	Internal signal from on-chip timer
dac_chx_trg5	tim9_trgo (DAC1, 5, 7, 9) tim12_trgo (DAC2, 6, 8, 10)	Internal signal from on-chip timer
dac_chx_trg6	tim15_trgo	Internal signal from on-chip timer
dac_chx_trg7	tim20_trgo	Internal signal from on-chip timer
dac_chx_trg8	reserved	-
dac_chx_trg9	tim26_trgo	Internal signal from on-chip timer
dac_chx_trg10	tim27_trgo	Internal signal from on-chip timer
dac_chx_trg11	lptim_ch1	Internal signal from on-chip timer

Table 250. DAC interconnection (continued)

Signal name	Source	Source type
dac_chx_trg12	exti9	External pin
dac_chx_trg13	play1_out0 (DAC1) play1_out1 (DAC2) play1_out2 (DAC5) play1_out3 (DAC6) play1_out4 (DAC7) play1_out5 (DAC8) play1_out6 (DAC9) play1_out7 (DAC10)	Internal signal from PLAY
dac_chx_trg14	hrtim1_dac_trg1	Internal signal from on-chip timer
dac_chx_trg15	hrtim1_dac_trg2	Internal signal from on-chip timer
dac_chx_trg16	htrim1_dac_trg3	Internal signal from on-chip timer
dac_chx_trg17	hrtim1_dac_rst_trg1	Internal signal from on-chip timer
dac_chx_trg18	hrtim1_dac_rst_trg2	Internal signal from on-chip timer
dac_chx_trg19	hrtim1_dac_rst_trg3	Internal signal from on-chip timer
dac_chx_trg20	hrtim1_dac_rst_trg4	Internal signal from on-chip timer
dac_chx_trg21	hrtim1_dac_rst_trg5	Internal signal from on-chip timer
dac_chx_trg22	hrtim1_dac_rst_trg6	Internal signal from on-chip timer
dac_chx_trg23	reserved	-
dac_chx_trg24	reserved	-
dac_chx_trg25	reserved	-
dac_chx_trg26	reserved	-
dac_chx_trg27	reserved	-
dac_chx_trg28	reserved	-
dac_chx_trg29	reserved	-
dac_chx_trg30	reserved	-
dac_chx_trg31	reserved	-
dac_inc_chx_trg1	tim1_trgo	Internal signal from on-chip timer
dac_inc_chx_trg2	tim1_trgo2	Internal signal from on-chip timer
dac_inc_chx_trg3	tim2_trgo (DAC1, 7) tim3_trgo (DAC2, 8) tim4_trgo (DAC55, 9) tim5_trgo (DAC6, 10)	Internal signal from on-chip timer
dac_inc_chx_trg4	tim6_trgo	Internal signal from on-chip timer
dac_inc_chx_trg5	tim7_trgo	Internal signal from on-chip timer
dac_inc_chx_trg6	tim8_trgo	Internal signal from on-chip timer

Table 250. DAC interconnection (continued)

Signal name	Source	Source type
dac_inc_chx_trg7	tim8_trgo2	Internal signal from on-chip timer
dac_inc_chx_trg8	tim12_trgo (DAC1, 3, 7, 9) tim9_trgo (DAC2, 6, 8, 10)	Internal signal from on-chip timer
dac_inc_chx_trg9	tim15_trgo	Internal signal from on-chip timer
dac_inc_chx_trg10	tim20_trgo	Internal signal from on-chip timer
dac_inc_chx_trg11	tim20_trgo2	Internal signal from on-chip timer
dac_inc_chx_trg12	reserved	-
dac_inc_chx_trg13	reserved	-
dac_inc_chx_trg14	tim26_trgo	Internal signal from on-chip timer
dac_inc_chx_trg15	tim27_trgo	Internal signal from on-chip timer
dac_inc_chx_trg16	lptim_ch1	Internal signal from on-chip timer
dac_inc_chx_trg17	reserved	-
dac_inc_chx_trg18	exti10	External pin
dac_inc_chx_trg19	play1_out8 (DAC1) play1_out9 (DAC2) play1_out10 (DAC5) play1_out11 (DAC6) play1_out12 (DAC7) play1_out13 (DAC8) play1_out14 (DAC9) play1_out15 (DAC10)	Internal signal from PLAY
dac_inc_chx_trg20	hrtim1_dac_step_trg_out1	Internal signal from on-chip timer
dac_inc_chx_trg21	hrtim1_dac_step_trg_out2	Internal signal from on-chip timer
dac_inc_chx_trg22	hrtim1_dac_step_trg_out3	Internal signal from on-chip timer
dac_inc_chx_trg23	hrtim1_dac_step_trg_out4	Internal signal from on-chip timer
dac_inc_chx_trg24	hrtim1_dac_step_trg_out5	Internal signal from on-chip timer
dac_inc_chx_trg25	hrtim1_dac_step_trg_out6	Internal signal from on-chip timer
dac_inc_chx_trg26	reserved	-
dac_inc_chx_trg27	reserved	-
dac_inc_chx_trg28	reserved	-
dac_inc_chx_trg29	reserved	-
dac_inc_chx_trg30	reserved	-
dac_inc_chx_trg31	reserved	-

28.4.3 DAC channel enable

The DAC channel can be powered on by setting its corresponding EN1 bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The EN1 bit enables the analog DAC channel1 only. The DAC channel1 digital interface is enabled even if the EN1 bit is reset.

28.4.4 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

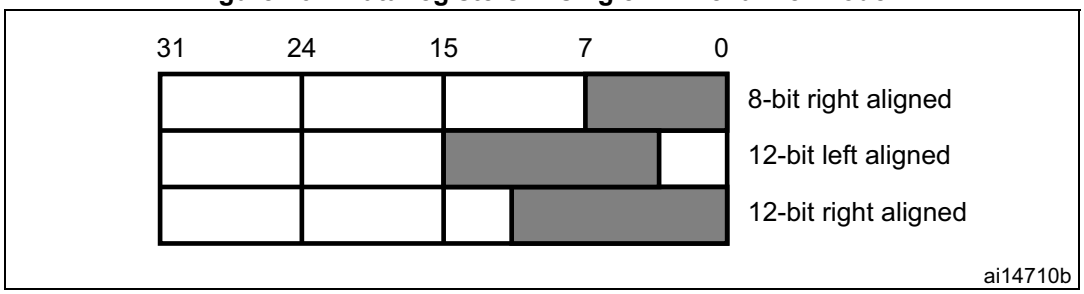
  – 8-bit right alignment: the software has to load data into the DAC_DHR8R1[7:0] bits (stored into the DHR1[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12L1[15:4] bits (stored into the DHR1[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12R1[11:0] bits (stored into the DHR1[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DAC_DHR1 (data holding registerx, which are internal non-memory-mapped registers). The DAC_DHR1 register is then loaded into the DAC_DOR1 register either automatically, by software trigger or by an external event trigger.

Figure 267. Data registers in single DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2’s complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB of the data written to DAC_DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB is inverted. The four LSBs are simply ignored.

Table 251. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DAC_DHRx register	DATA transfered to DAC_DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

28.4.5 DAC conversion

The DAC_DOR1 cannot be written directly and any data transfer to the DAC channel1 must be performed by loading the DAC_DHR1 register (write operation to DAC_DHR8R1, DAC_DHR12L1, DAC_DHR12R1).

Data stored in the DAC_DHR1 register are automatically transferred to the DAC_DOR1 register after one dac_ker_ck clock cycle, if no hardware trigger is selected (TEN1 bit in DAC_CR register is reset). However, when a hardware trigger is selected (TEN1 bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_ker_ck clock cycles after the trigger signal.

When DAC_DOR1 is loaded with the DAC_DHR1 contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

HFSEL bits of DAC_MCR must be set when dac_ker_ck clock speed is faster than 80 MHz.

Refer to Table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and hclk or dac_ker_ck clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

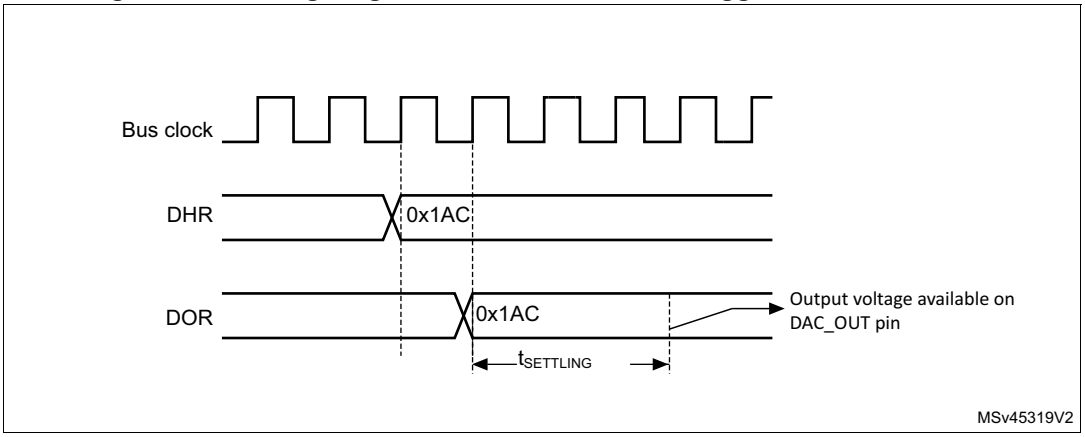
The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 252. HFSEL description

HFSEL[2:0]	dac_ker_ck frequency	Latency using dac_ker_ck clock	Function
000	< 80 MHz	4	DAC_DOR update rate up to 4 dac_ker_ck cycles.
001	≥ 80 MHz(1)	5	DAC_DOR update rate up to 5 dac_ker_ck cycles.
010	≥ 160 MHz	6	DAC_DOR update rate up to 6 dac_ker_ck cycles.

1. Refer to the device datasheet for the value of the maximum kernel clock frequency.

Figure 268. Timing diagram for conversion with trigger disabled TEN = 0



28.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between $V_{REFDAC-}$ and $V_{REFDAC+}$.

The analog output voltage on the DAC channel pin is determined by the following equation:

$$\text{DAC output} = V_{REFDAC} \times \frac{DOR}{4096}$$

where all voltages are expressed in Volt.

$$V_{REFDAC} = V_{REFDAC+} - V_{REFDAC-}$$

28.4.7 DAC trigger selection

If the TEN1 control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSEL1[4:0] control bits determine which out of 32 possible events triggers the conversion as shown in TSEL1[4:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers (see the interconnection table in *Section 28.4.2*).

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHR1 register are transferred into the DAC_DOR1 register. The DAC_DOR1 register is updated three dac_ker_ck cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DOR1 register has been loaded with the DAC_DHR1 register contents.

The reset trigger selection and the increment trigger selection of the sawtooth generation are performed through STRSTTRIGSELx and STINCTRIGSELx control bits, respectively. STRSTTRIGSELx mapping is similar to TSELx. Refer to the *Section 28.4.2: DAC pins and internal signals* for TSELx, STRSTTRIGSELx, and STINCTRIGSELx mappings.

Note: TSEL1[4:0] bit cannot be changed when the EN1 bit is set.

When software trigger is selected, the transfer from the DAC_DHR1 register to the DAC_DOR1 register takes only one dac_ker_ck clock cycle.

28.4.8 DMA requests

The DAC channel has a DMA capability. One DMA channel is used to service DAC channel DMA request.

When an external trigger (but not a software trigger) occurs while the DMAEN1 bit is set, the value of the DAC_DHR1 register is transferred into the DAC_DOR1 register when the transfer is complete, and a DMA request is generated.

As DAC_DHR1 to DAC_DOR1 data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHR1 before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channel1 underrun flag DMAUDR1 in the DAC_SR register is set, reporting the error condition. The DAC channel1 continues to convert old data.

The software must clear the DMAUDR1 flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channel1 to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For DAC channel1, an interrupt is also generated if its corresponding DMAUDRIE1 bit in the DAC_CR register is enabled.

DMA double data mode

When the DMA controller is used in normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA double data mode.

The following conditions must be met to change from double data to single data mode or vice versa:

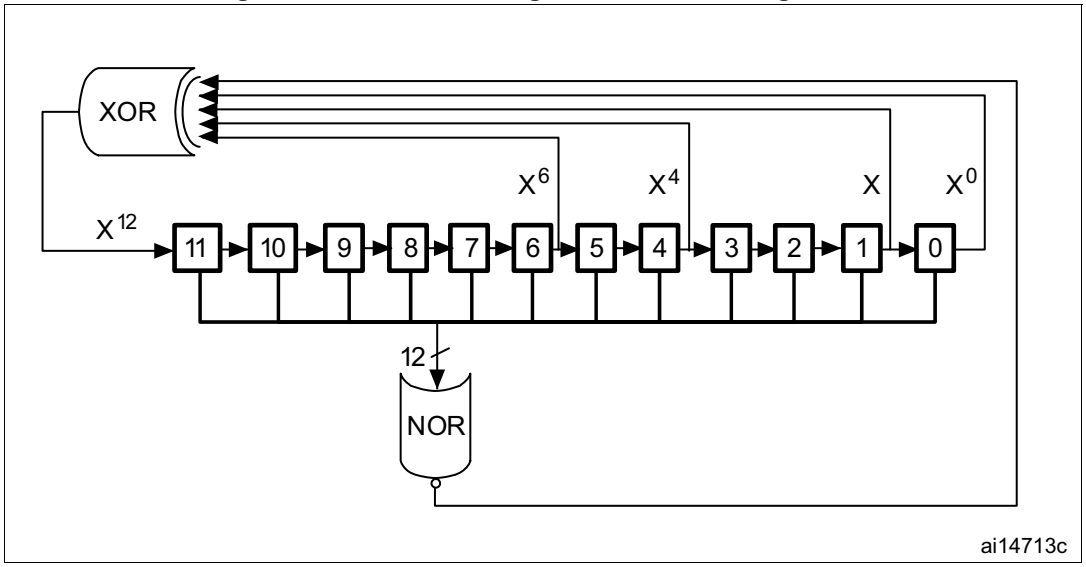
- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

28.4.9 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVE1[1:0] to 01. The

preloaded value in LFSR is 0xAAA. This register is updated three dac_ker_ck clock cycles after each trigger event, following a specific calculation algorithm.

Figure 269. DAC LFSR register calculation algorithm

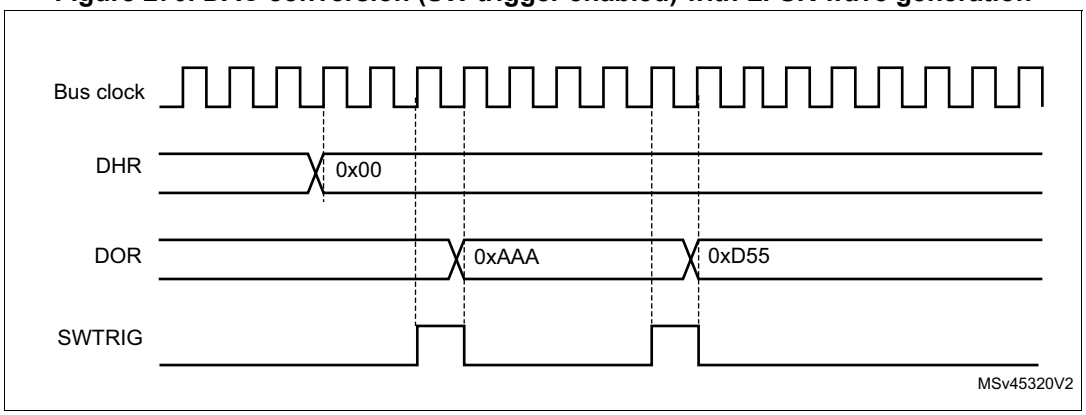


The LFSR value, that may be masked partially or totally by means of the MAMP1[3:0] bits in the DAC_CR register, is added up to the DAC_DHR1 contents without overflow and this value is then transferred into the DAC_DOR1 register.

If LFSR is 0x0000, a 1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVE1[1:0] bits.

Figure 270. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TEN1 bit in the DAC_CR register.

28.4.10 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVE1[1:0] to 10. The amplitude is configured through the MAMP[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_ker_ck clock cycles after each trigger event. The value of this counter is then added to the DAC_DHR1 register without overflow and the sum is transferred into the DAC_DOR1 register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMP1[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVE1[1:0] bits.

Figure 271. DAC triangle wave generation

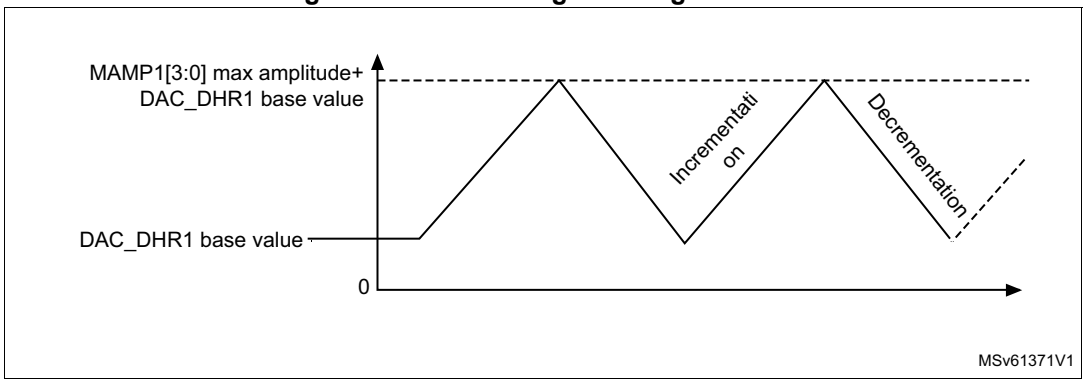
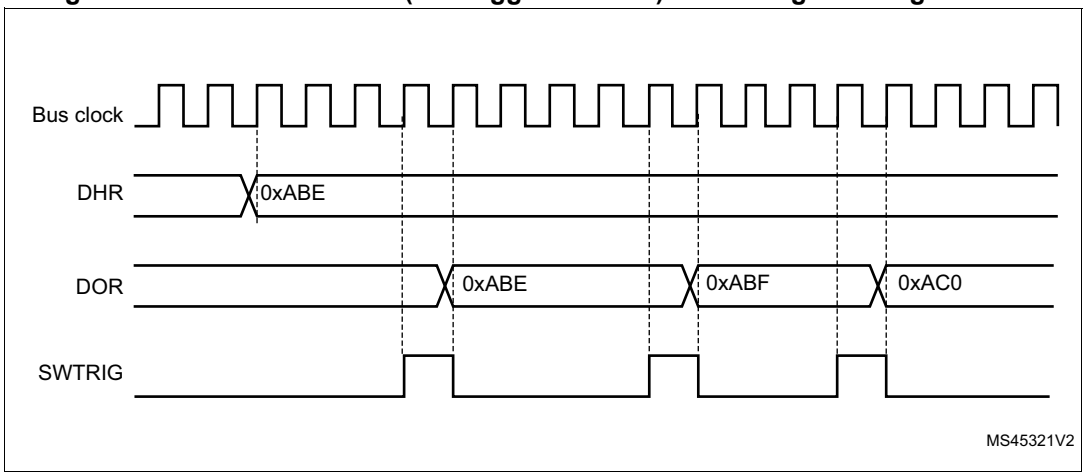


Figure 272. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TEN1 bit in the DAC_CR register.

The MAMP1[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

28.4.11 DAC sawtooth wave generation

The DAC can generate a sawtooth waveform. Specific register settings for the initial value, increment value and direction control are required:

- DAC sawtooth wave generation is selected by setting WAVEx[1:0] to 11 in the DAC_CR register.
- The sawtooth counter initial value (reset value) is configured through STRSTDATAx[11:0] bits in the DAC_STRx register.
- The increment value is defined by the STINCDATAx[15:0] bits in the DAC_STRx register.
- The sawtooth direction is defined by STDIRx bit in the DAC_STRx register.

The sawtooth counter starts from STRSTDATAx[11:0] (bits 12 to 15 are set to 0b0000), each increment trigger then increments (or decrements) STINCDATAx[15:0] value.

The DAC output is used from 12 MSBs of those counter value. When the counter reaches 0x0000 or 0xFFFF, the value is saturated. The sawtooth reset trigger signal initializes the counter value to the STRSTDATAx[11:0] (bits 12 to 15 are set to 0b0000) value.

The increment trigger (STINCTRIG) and reset trigger (STRSTTRIG) must be selected through the STINCTRIGSELx[3:0] and the STRSTTRIGSELx[3:0] bits.

Figure 273. DAC sawtooth wave generation (STDIRx = 0)

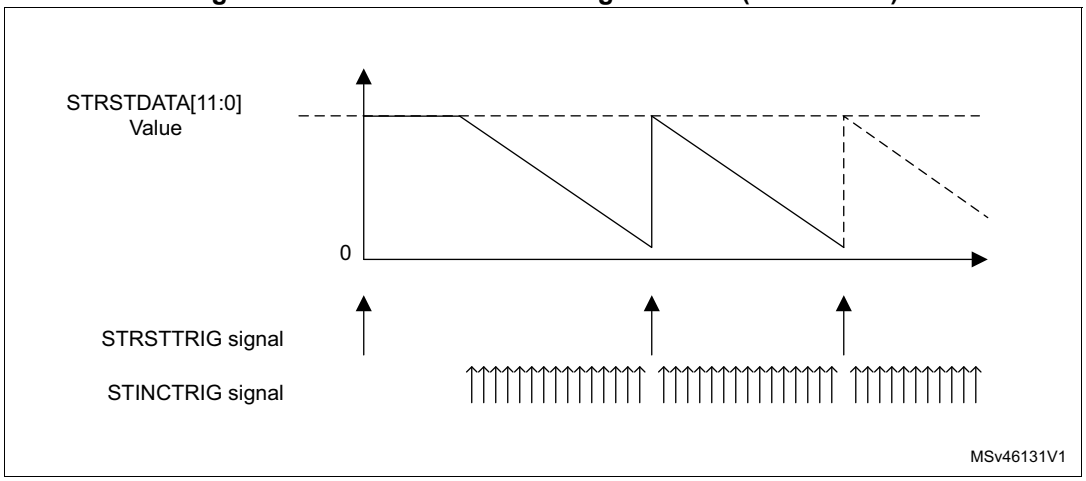
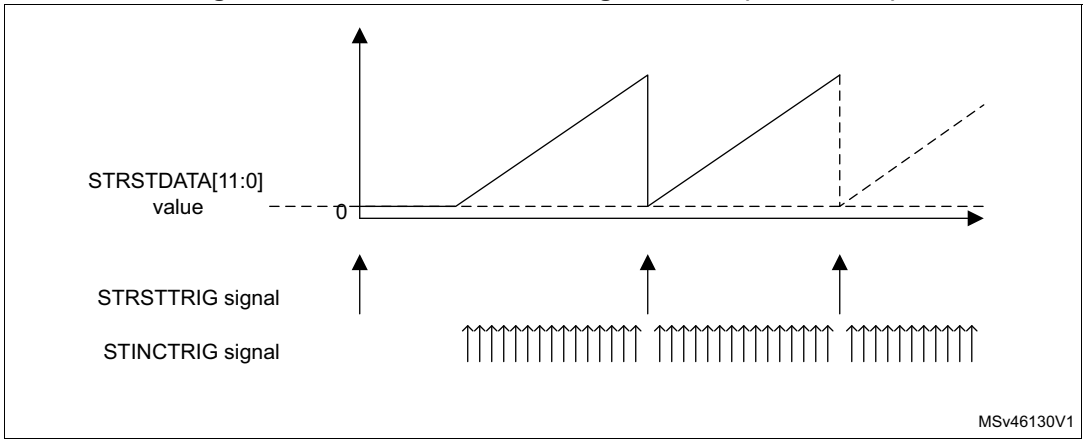
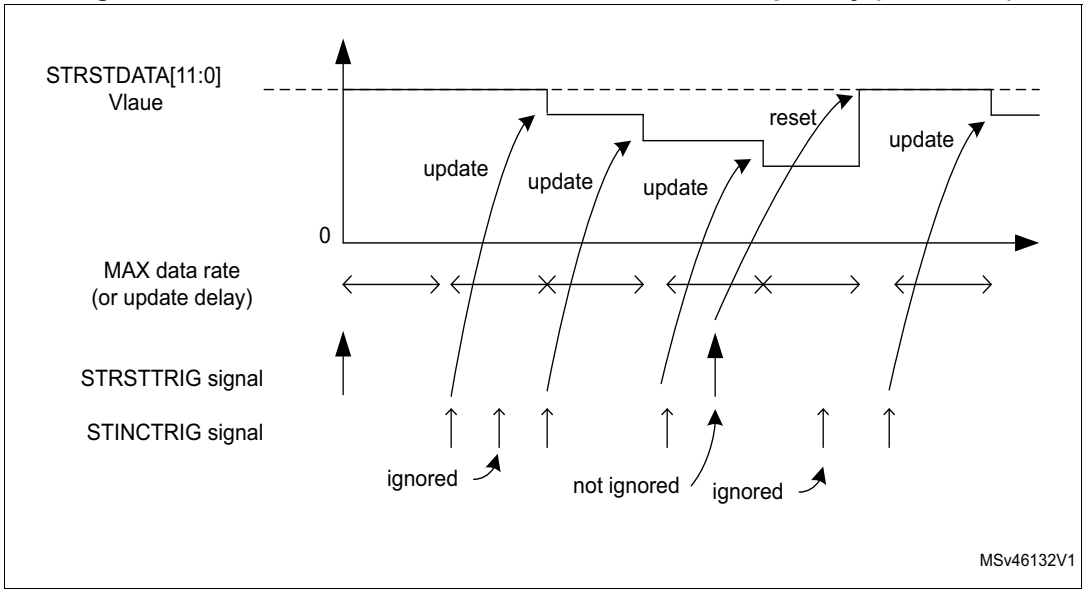


Figure 274. DAC sawtooth wave generation (STDIRx = 1)



The STRSTTRIG signal has higher priority than STINCTRIG. The trigger signal cannot be faster than the DAC_DORx update rate (see *Table HFSEL description - if available - for details)*. If STINCTRIG is asserted faster than the allowed data update rate, the STINCTRIG trigger is ignored. If the STRSTTRIG signal is applied after the STINCTRIG and before DAC_DORx update rate constraints, STRSTTRIG is put on hold. Then, immediately after the data increment, the reset trigger is applied.

Figure 275. DAC sawtooth STINCTRIG and STRSTTRIG priority (STDIR = 0)



28.4.12 DAC extended sawtooth wave generation

The extended sawtooth wave generation is more flexible than the standard sawtooth wave generation described in *Section 28.4.11: DAC sawtooth wave generation*:

- It is possible to define the minimum and maximum output values.
- The extended sawtooth is managed by a signed 20-bit internal counter, enabling to convert data in 16.4 format.

  The first MSB is the sign bit, the second to the forth MSBs contain the overflow part, the middle 12 bits (5 to 16 MSBs) are used for the DAC output data, and the four LSBs consist of the fractional part. When the four MSBs equal 0b0000, the counter value is transferred to the DAC output. Otherwise, the DAC output value is saturated at 0x000, 0xFFF, or at the minimum or maximum value, programmed in the DAC_STEXTMM1 register.

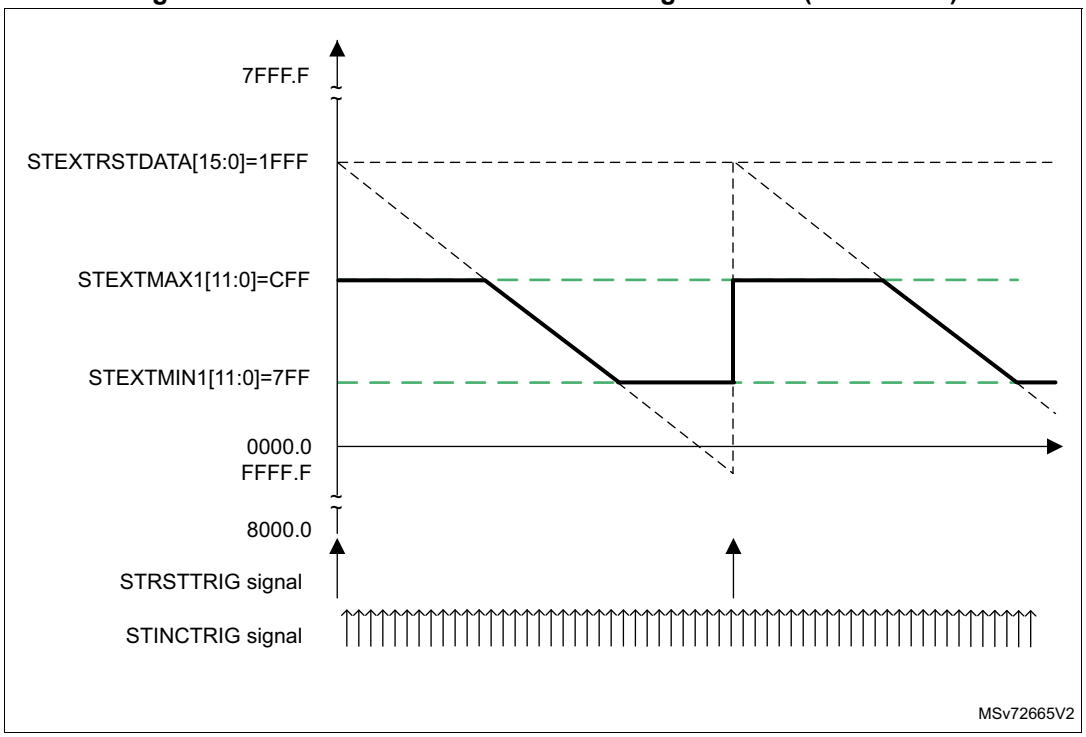
The following sequence is required to configure and start the extended sawtooth generation:

1. Clear the EN1 bit of the DAC_CR register.
2. Activated the extended sawtooth mode by setting the STEXT1 bit of the DAC_STR1 register.
3. Configure the sawtooth counter initial value (reset value) through the STEXTRSTDATA1[15:0] bitfield of the DAC_STEXTRST register.
4. Configure the increment value through the STINCDATA1[15:0] bitfield of the DAC_STRx register whose four LSBs correspond to the fractional part.
5. Configure the sawtooth minimum and maximum values through the STEXTMIN1[11:0] and STEXTMAX1[11:0] bitfield of the DAC_STEXTMM1 register
6. Configure the sawtooth direction through the STDIR1 bit of the DAC_STR1 register.
7. Select the sawtooth reset and increment trigger through the STRSTTRIGSEL1[3:0] and STINCTRIGSEL1[3:0] bitfields.
8. Set WAVE1[1:0] to 0b11 and EN1 to 1.
9. Generate a sawtooth reset trigger to start the sawtooth generation.

When STDIR1 is cleared (decrement case), the internal counter operates as follows:

- When the STRSTTRIG is applied, the counter 16 MSBs are reset to the signed 20-bit STEXTRSTDATA1[15:0] value, and the four LSBs are reset to 0b0000.
- When the STINCTRIG is applied, the counter is subtracted from STINCDATA1[15:0].
- When the counter four MSBs are greater than or equal to 0b0001, the DAC output is saturated at the STEXTMAX1[11:0] value.
- When the counter four MSBs are equal to 0b0000, the counter middle 12-bit value is evaluated. If it is lower than STEXTMAX1[11:0] and greater than STEXTMIN1[11:0], the value is transferred to the DAC output.
- When the counter middle 12-bit value is lower than STEXTMIN1[11:0], the DAC output value is saturated at STEXTMIN1[11:0].
- When the counter value reaches 0x0000.0, the counter continues down-counting below the negative value 0xFFFFF. If the value reaches 0x8000.0, the internal counter value is saturated at 0x8000.0.

Figure 276. DAC extended sawtooth wave generation (STDIR1 = 0)

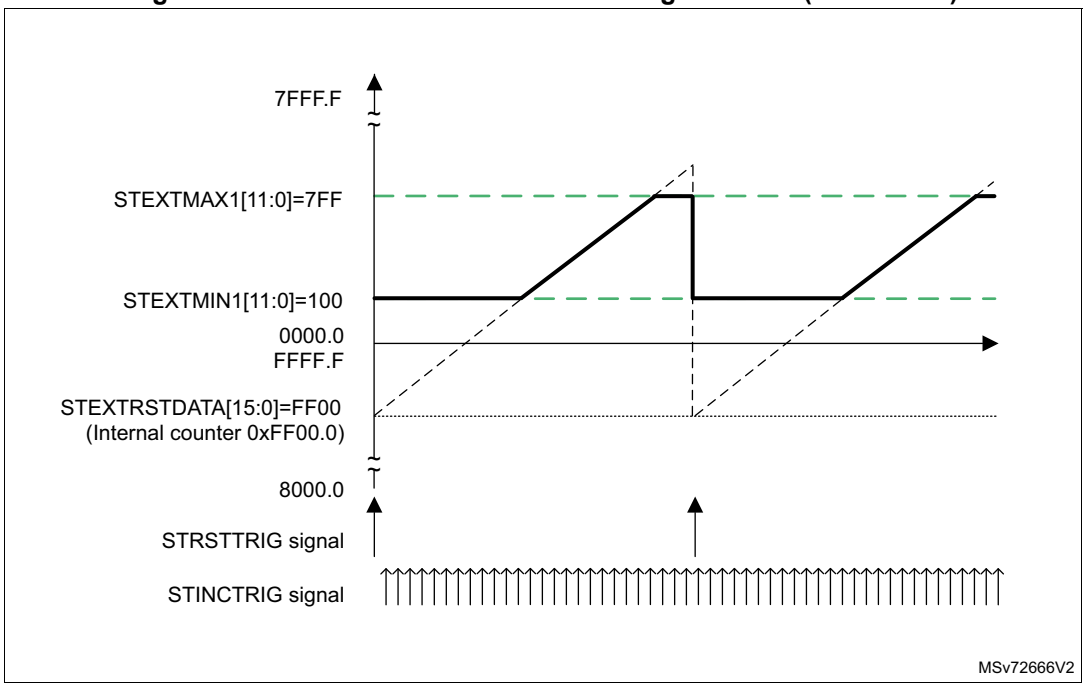


1. The line in bold represents the DAC output, the dashed line in black the internal counter value, and the dashed lines in green the extended sawtooth minimum and maximum values.

When STDIR1 is set (increment case), the internal counter operates as follows:

- When the STRSTTRIG is applied, the counter 16 MSBs are reset to STEXTRSTDATA1[15:0] value, and the four LSBs are reset to 0x0.
- When the STINCTRIG is applied, the counter is added to STINCDATA1[15:0].
- When the counter is negative (MSB = 1), the DAC output is saturated at the STEXTMIN1[11:0] value.
- When the counter four MSBs are equal to 0b0000, the counter middle 12-bit value is evaluated. If it is lower than STEXTMAX1[11:0] and greater than STEXTMIN1[11:0], the value is transferred to the DAC output.
- When the counter middle 12-bit value is greater than the STEXTMAX1[11:0] value, the DAC output value is saturated at STEXTMAX1[11:0].
- When the value reaches 0x7FFF.F, the counter is saturated.

Figure 277. DAC extended sawtooth wave generation (STDIR1 = 1)



1. The line in bold represents the DAC output, the dashed line in black the internal counter value, and the dashed lines in green the extended sawtooth minimum and maximum values.

Dynamic modification of the extended sawtooth parameters

The STEXTRSTDATA1[15:0] bitfield can be modified while an extended sawtooth generation is ongoing. The new value is taken into account when a new STRSTTRIG is triggered.

When STINCDATA1[15:0] and/or STDIR1 are modified, the new value is applied when a new STINCTRIG is triggered.

When either or both of STEXTMAX1[11:0] and STEXTMIN1[11:0] are modified, the new value is applied when the next STRSTTRIG or STINCTRIG is triggered. To ensure STEXTMAX1[11:0] is always greater than STEXTMIN1[11:0], it is recommended to program them simultaneously through the DAC_STEXTMM1 register.

Since all the extended sawtooth parameters are stored in registers located in the AHB domain, and the DAC operates from the kernel clock, if an STRSTTRIG or an STINCTRIG signal arrives while a sawtooth parameter is being updated, the old value may be taken into account instead of the new one. To prevent this issue from occurring, the extended sawtooth parameters must be written much before the next trigger.

28.4.13 DAC channel modes

The DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUT1 pin interconnections.

To enable the output buffer, the MODE1[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODE1[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI/LSE low-speed clock (dac_hold_ck) in addition to the dac_ker_ck clock, allowing using the DAC channel in deep low power modes such as Stop mode.

The LSI/LSE low-speed clock (dac_hold_ck) must not be stopped when the sample and hold mode is enabled.

The sample/hold mode operations can be divided into three phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLE1[9:0] bits in DAC_SHSR1 register. During the write of the TSAMPLE1[9:0] bits, the BWST1 bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLD1[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESH1[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI/LSE clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI/LSE ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLE1[9:0] = 11.
- 62 cycles are required for hold phase: THOLD1[9:0] = 62.
- and 4 cycles are required for refresh period: TREFRESH1[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 253. Sample and refresh timings

Buffer State	t_{SAMP}(1)(2)	$t_{REFRESH}$(2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula, the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8-bit resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only (refer to the datasheet for product data):

C_{SH} = 100 nF

$V_{REFDAC+}$ = 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms

(where R_{BON} = 2 kΩ)

Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs

(where N_{LSB} = 10 (10 LSB drop during the hold phase)

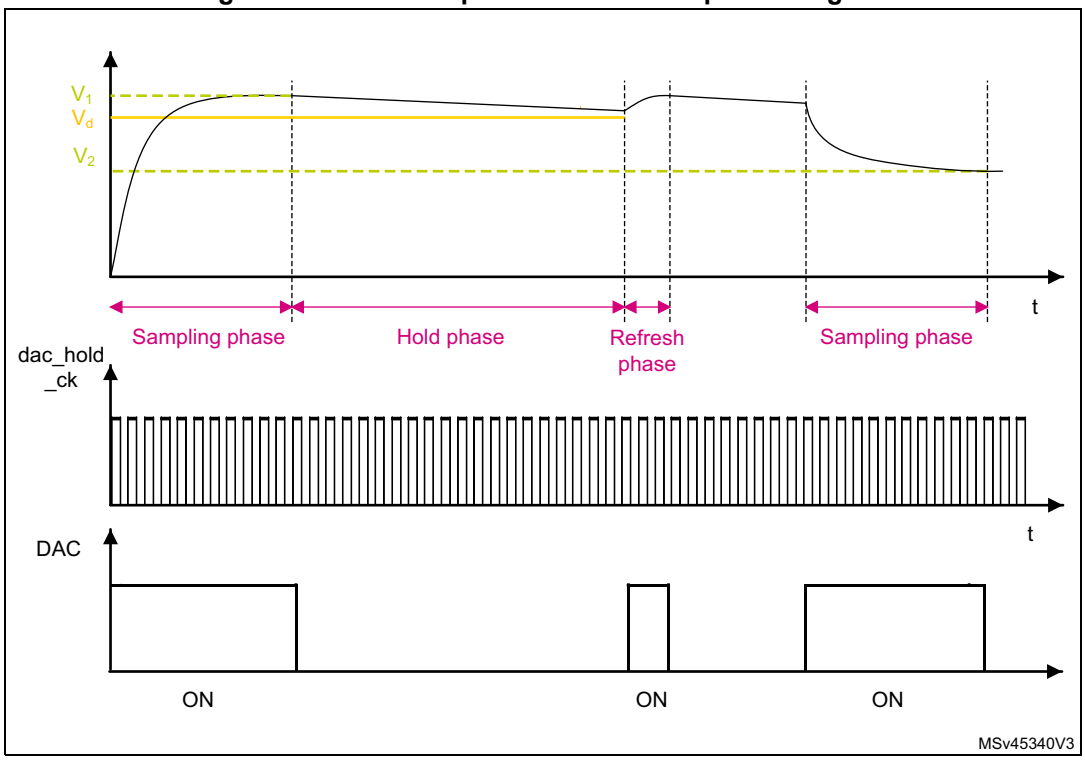
Hold phase:

D_v = i_{leak} * t_{hold} / C_{SH} = 0.0073 V (10 LSB of 12bit at 3 V)

i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)

t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 278. DAC sample and hold mode phase diagram



Like in normal mode, the sample and hold mode has different configurations.

To enable the output buffer, MODE1[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODE1[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

  When MODE1[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All sample and hold phases are interruptible, and any change in DAC_DHR1 immediately triggers a new sample phase.

Note: The sawtooth wave generation is not supported in sample and hold mode operation.

Table 254. Channel output modes summary

MODE1[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)

Table 254. Channel output modes summary (continued)

MODE1[2:0]			Mode	Buffer	Output connections
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

28.4.14 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = ((D / 2^N) \times G \times V_{REFDAC}) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{REFDAC} is the nominal full-scale voltage, and V_{OS} is the offset voltage. For an ideal DAC channel, G = 1 and V_{OS} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{OS}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channel is operating with buffer enabled (MODE1[2:0] = 0b000 or 0b001 or 0b100 or 0b101). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to V_{REFDAC}/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAG1 bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIM1[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, V_{REFDAC} values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the nominal factory trimming conditions.

In addition, when V_{DD} is removed (example the device enters in Standby mode) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to EN1 bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODE1[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channel calibration, by setting the CEN1 bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIM1[4:0] bits, starting by 0b00000.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAG1 bit in DAC_SR is set to 1.
   d) Until the CAL_FLAG1 is read as 1 or the maximum trimming code is reached, increment OTRIM1[4:0] and repeat substeps from (b) to (d).

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIM1[4:0] bits in a faster way.

Note: A t_{TRIM} delay must be respected between the write to the OTRIM1[4:0] bits and the read of the CAL_FLAG1 bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA}, V_{REFDAC} and temperature conditions do not change during device operation while it enters more often in Standby mode, the software may store the OTRIM1[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CEN1 bit is set, it is not allowed to set EN1 bit.

28.4.15 DAC channel conversion modes

Four conversion modes are possible.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[4:0] bits.
3. Load the DAC channel data into the desired DHR registers (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[4:0] bits.
3. Configure the DAC channel WAVE1[1:0] bits as 01 and the same LFSR mask value in the MAMP1[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). Then the LFSR1 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, TEN1.
2. Configure the trigger sources by setting different values in the TSEL1[4:0] bits.
3. Configure the DAC channel WAVE1[1:0] bits as 1x and the same maximum amplitude value in the MAMP1[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_ker_ck clock cycles later). The DAC channel triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the trigger sources by setting different values in STRSTTRIGSELx[3:0] and STINCTRIGSELx[3:0] bits.
2. Configure the DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx values for each register.

When a DAC channel trigger arrives, the DAC channel sawtooth counter updates the DHRx register and transfers it into DAC_DOR1 (three AHB clock cycles later).

28.5 DAC in low-power modes

Table 255. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA.
Stop(1)	The DAC remains active with a static value. The sample and hold mode can be selected using LSE/LSI clock.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 28.3: DAC implementation* for information on the Stop modes supported by the DAC peripheral.

28.6 DAC interrupts

Table 256. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDR1	DMAUDRIE1	Write DMAUDRx = 1	Yes	No	No

28.7 DAC registers

Refer to *Section 1* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

28.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSEL1[4]	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:0]				TEN1	EN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.
00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Sawtooth wave generation enabled
Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 15, 5:2 **TSEL1[4:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1
00000: SWTRIG1
00001: dac_ch1_trg1
00010: dac_ch1_trg2
...
11111: dac_ch1_trg31
For details on trigger configuration and mapping, refer to *Section 28.4.2: DAC pins and internal signals*.

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_ker_ck clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_ker_ck clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_ker_ck clock cycle.*

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

28.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIGB1
															w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG1
															w

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SWTRIGB1**: DAC channel1 software trigger B

This bit is set by software to trigger the DAC in software trigger mode (sawtooth generation) It is cleared by hardware.

0: No trigger

1: Trigger for sawtooth increment

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode, or software reset trigger mode for the sawtooth wave generation.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_ker_ck clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

28.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel1.

28.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC1DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

28.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHRB[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

28.7.6 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

28.7.7 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag
This bit is systematically set just after sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI/LSE periods of synchronization).
0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written
1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status
This bit is set and cleared by hardware
0: calibration trimming value is lower than the offset correction value
1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

28.7.8 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

28.7.9 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL[2:0]			Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw	rw				rw	rw						rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **HFSEL[2:0]**: High frequency interface mode selection
000: High frequency interface mode disabled
001: High frequency interface mode enabled for 80 MHz ≤ dac_ker_ck ≤ 160 MHz
010: High frequency interface mode enabled for 160 MHz ≤ dac_ker_ck
≤≤≤≤Others: Reserved, must not be used.

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA normal mode selected
1: DMA double data mode selected

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in normal mode
000: DAC channel1 is connected to external pin with Buffer enabled
001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
010: DAC channel1 is connected to external pin with Buffer disabled
011: DAC channel1 is connected to on chip peripherals with Buffer disabled
– DAC channel1 in sample & hold mode
100: DAC channel1 is connected to external pin with Buffer enabled
101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
111: DAC channel1 is connected to on chip peripherals with Buffer disabled
Note: This register can be modified only when EN1 = 0.

28.7.10 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

28.7.11 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in sample and hold mode)
Hold time = (THOLD[9:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

28.7.12 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

28.7.13 DAC channel1 sawtooth register (DAC_STR1)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STINCDATA1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	STEXT1	STDIR1	STRSTDATA1[11:0]											
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **STINCDATA1[15:0]:** DAC channel1 sawtooth increment value (12.4 bit format)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **STEXT1**: DAC channel1 sawtooth extended setting
0: Sawtooth reset value defined by STRSTDATA[11:0]
1: Sawtooth reset value defined by STEXTRSTDATA[15:0]

Bit 12 **STDIR1:** DAC channel1 sawtooth direction setting
This bit is written by software to select the direction of sawtooth step direction:
0: Decrement
1: Increment

Bits 11:0 **STRSTDATA1[11:0]:** DAC channel1 sawtooth reset value

28.7.14 DAC sawtooth mode register (DAC_STMODR)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STINCTRIGSEL1[4:0]					Res.	Res.	Res.	STRSTTRIGSEL1[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **STINCTRIGSEL1[4:0]:** DAC channel1 sawtooth increment trigger (STINCTRIG) selection
For details on trigger configuration and mapping, refer to *Section 28.4.2: DAC pins and internal signals*.
00000: SWTRIGB1
00001: dac_inc_ch1_trg1
....
11111: dac_inc_ch1_trg31

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **STRSTTRIGSEL1[4:0]:** DAC channel1 sawtooth reset trigger (STRSTTRIG) selection
For details on trigger configuration and mapping, refer to *Section 28.4.2: DAC pins and internal signals*.
00000: SWTRIG1
00001: dac_ch1_trg1
....
11111: dac_ch1_trg31
The mapping is the same as for TSEL1[4:0].

28.7.15 DAC extended sawtooth reset control register (DAC_STEXTRST)

Address offset: 0x64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STEXTRSTDATA1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **STEXTRSTDATA1[15:0]:** DAC channel1 extended sawtooth reset value
This bitfield contains the extended sawtooth value programmed at reset, in 2's complement signed format.

28.7.16 DAC extended sawtooth minimum and maximum value register (DAC_STEXTMM1)

Address offset: 0x68

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STEXTMAX1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	STEXTMIN1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **STEXTMAX1[11:0]:** DAC channel1 extended sawtooth maximum value
This bitfield contains the extended sawtooth maximum out value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **STEXTMIN1[11:0]:** DAC channel1 extended sawtooth minimum value
This bitfield contains the extended sawtooth minimum out value.

28.7.17 DAC option register (DAC_OR)

Address offset: 0x3EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **OP[1:0]:** Option bits
For DAC5, 6, 7, 8, 9, 10
00: Output glitch control disabled
01: Output glitch control forced to the reverse direction
10: Output glitch control forced to the forward direction
11: Reserved, must not be used.

28.7.18 DAC register map

Table 257 summarizes the DAC registers.

Table 257. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	Res.	Res.	Res.	Res.				Res.		Res.				Res.	Res.	TSEL1[4]	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:0]				TEN1	EN1
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_ SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIGB1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG1
	Reset value																0																0
0x08	DAC_ DHR12R1	Res.	Res.	Res.	Res.	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 257. DAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	DAC_DHR12L1	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHRB[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14 - 0x1C	Reserved																																
0x20 - 0x28	Reserved																																
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	DACC1DORB[11:0]												Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	DAC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0											
0x38	DAC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4]	OTRIM1[3]	OTRIM1[2]	OTRIM1[1]	OTRIM1[0]
	Reset value																												X	X	X	X	X
0x3C	DAC_MCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HFSEL[2]	HFSEL[1]	HFSEL[0]	Res.	es.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
	Reset value																	0	0	0				0	0						0	0	0
0x40	DAC_SHSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																
0x48	DAC_SHHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	1
0x4C	DAC_SHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0x50-0x54	Reserved																																
0x58	DAC_STR1	STINCDATA1[15:0]																Res.	Res.	STEXT1	STDIR1	STRSTDATA1[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	Reserved	Res.																															

Table 257. DAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	DAC_ STMODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STINCTRIGSEL1[4]	STINCTRIGSEL1[3]	STINCTRIGSEL1[2]	STINCTRIGSEL1[1]	STINCTRIGSEL1[0]	Res.	Res.	Res.	STRSTTRIGSEL1[4]	STRSTTRIGSEL1[3]	STRSTTRIGSEL1[2]	STRSTTRIGSEL1[1]	STRSTTRIGSEL1[0]
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x64	DAC_ST EXTRST	Res.																STEXTRSTDATA1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	DAC_ST EXTMM1	Res.	Res.	Res.	Res.	STEXTMAX1[11:0]												Res.	Res.	Res.	Res.	STEXTMIN1[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x6C	Reserved																																
0x3EC	DAC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP1	OP0
	Reset value																															0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

29 Voltage reference buffer (VREFBUF)

29.1 VREFBUF introduction

The devices embed a voltage reference buffer which can be used as voltage reference for the on-chip ADCs, and also as voltage reference for external components through the VREF+ pin.

29.2 VREFBUF implementation

The table below describes the VREFBUF voltage typical values:

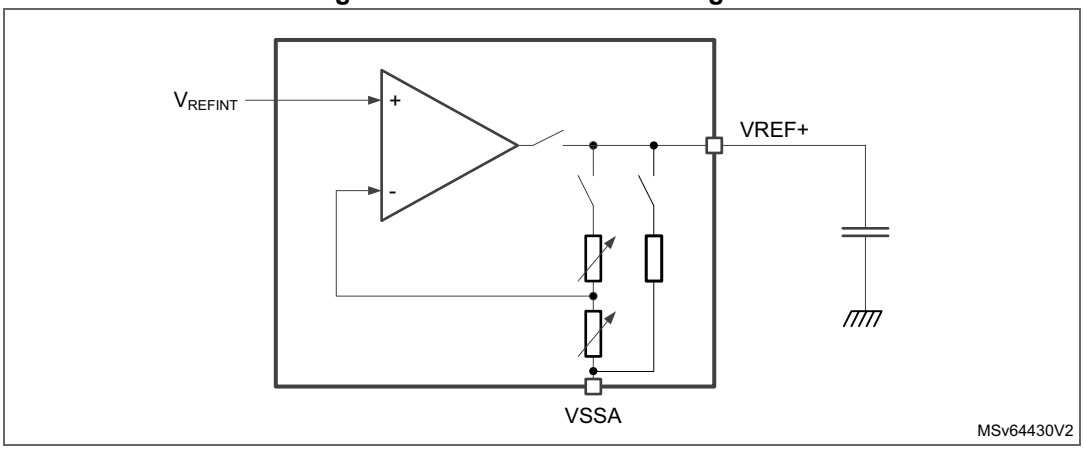
Table 258. VREFBUF typical values

Symbol	Value
$V_{REFBUF0}$	2.9 V
$V_{REFBUF1}$	2.5 V
$V_{REFBUF2}$	2.048 V
$V_{REFBUF3}$	1.8 V

Note: Refer to the product datasheet for more details.

29.3 VREFBUF functional description

Figure 279. VREFBUF block diagram



The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference V_{REFINT}. The VREFBUF supports four voltages[(a)], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: $V_{REFBUF0}$ voltage selected
- VRS = 001: $V_{REFBUF1}$ voltage selected
- VRS = 010: $V_{REFBUF2}$ voltage selected
- VRS = 011: $V_{REFBUF3}$ voltage selected

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 259. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – VREF+ pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – VREF+ pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – VREF+ pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

29.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

29.5 VREFBUF registers

29.5.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
									rw	rw	rw	r		rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

VRS = 000: $V_{REFBUF0}$ voltage selected.
VRS = 001: $V_{REFBUF1}$ voltage selected.
VRS = 010: $V_{REFBUF2}$ voltage selected.
VRS = 011: $V_{REFBUF3}$ voltage selected.
Others: Reserved

Note: Refer to the product datasheet for each $V_{REFBUFx}$ voltage setting value.
The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the VREF+ pin.
0: VREF+ pin is internally connected to the voltage reference buffer output.
1: VREF+ pin is high impedance.
Refer to *Table 259: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.
0: Internal voltage reference mode disable (external voltage reference mode).
1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

29.5.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]**: Trimming code

The TRIM code is a 6-bit unsigned data (minimum 000000, maximum 111111) that is set and updated according the mechanism described below.

Reset:

TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the flash memory during the production test.

VRS change:

TRIM[5:0] is automatically initialized with the trimming value (corresponding to VRS setting) stored in the flash memory during the production test.

Write in TRIM[5:0]:

User can modify the TRIM[5:0] with an arbitrary value. This is permanently disabling the control of the trimming value with VRS (until the device is reset).

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

29.5.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 260. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											X	X	X	X	X	X

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

30 Comparator (COMP)

30.1 COMP introduction

The device embeds up to eight comparators, which can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal
- Analog signal conditioning
- Cycle-by-cycle current control loop when combined with a PWM output from a timer

30.2 COMP main features

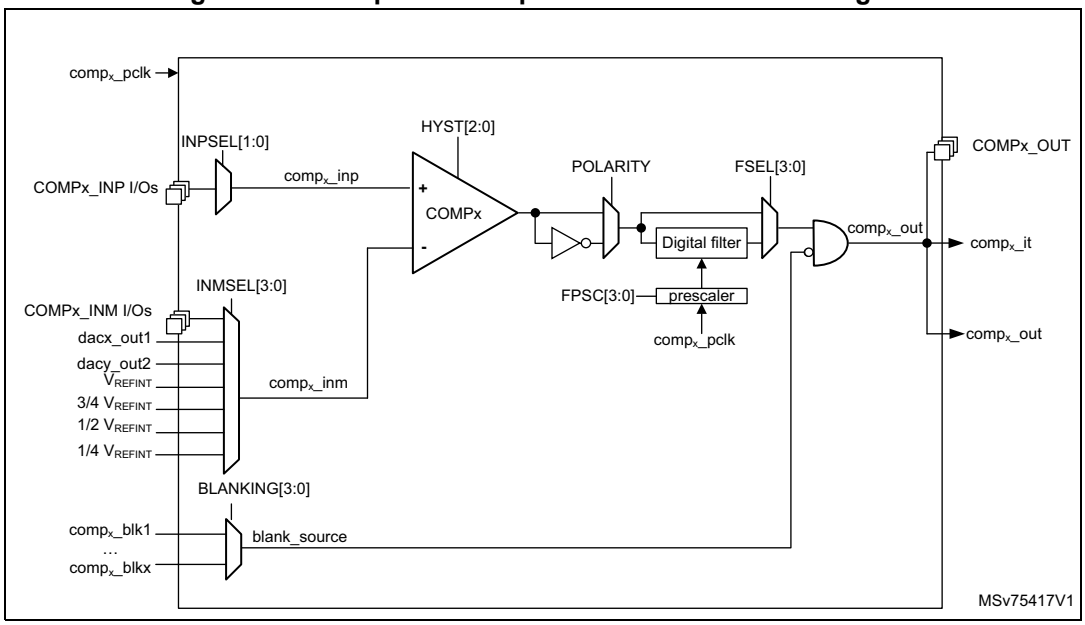
- Selectable inverting analog inputs:
  - Multiplexed I/O pins
  - DAC channels
  - Internal reference voltage and three sub-multiple values (1/4, 1/2, 3/4) provided by the scaler (buffered voltage divider)
- Programmable hysteresis
- Digital filtering
- Secured window mode
- Output redirection to I/Os or to timer inputs for triggering break events for fast PWM shutdowns
- Output blanking for immunity to switching noise
- Interrupt generation capability with wake up from Sleep and Stop modes (through the EXTI controller)

30.3 COMP functional description

30.3.1 COMP simplified block diagram

The simplified block diagram of the comparators is shown in the figure below.

Figure 280. Comparator simplified functional block diagram



30.3.2 COMP pins and internal signals

The I/Os used as comparator inputs must be configured in analog mode in the GPIO registers.

The comparator outputs can be connected to the I/Os through their alternate functions. Refer to the product datasheet for more information.

The outputs can also be internally redirected to a variety of timer inputs for the following purposes:

- emergency shutdown of PWM signals, using BKIN and BKIN2 inputs
- cycle-by-cycle current control, using ETR inputs of timers
- input capture for timing measurements

The comparator output can be routed simultaneously internally and to the I/O pins.

Table 261. COMPx_INP non-inverting input assignment (COMP1 to COMP8)(1)

INPSEL[1:0]	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
00	COMP1 INP1	COMP2 INP1	COMP3 INP1	COMP4 INP1	COMP5 INP1	COMP6 INP1	COMP7 INP1	COMP8 INP1
01	COMP1 INP2	COMP2 INP2	COMP3 INP2	COMP4 INP2	COMP5 INP2	COMP6 INP2	COMP7 INP2	COMP8 INP2

Table 261. COMPx_INP non-inverting input assignment (continued)(COMP1 to COMP8)(1)

INPSEL[1:0]	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
10	OPAMP1		OPAMP2		OPAM3		VSSA	
11	COMP2 INP	COMP1 INP	COMP4 INP	COMP3 INP	COMP6 INP	COMP5 INP	COMP8 INP	COMP7 INP

1. Other combinations are reserved.

Table 262. COMPx_INM inverting input assignment (COMP1 to COMP8)(1)

INMSEL[3:0]	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
0000	¼ V_{REFINT}							
0001	½ V_{REFINT}							
0010	¾ V_{REFINT}							
0011	V_{REFINT}							
0100	DAC1 INT	DAC2 INT	VSSA	VSSA	DAC5 INT	DAC6 INT	DAC7 INT	DAC8 INT
0101	DAC8 INT	DAC9 INT	DAC9 INT	DAC10 INT	DAC2 INT	VSSA	VSSA	VSSA
0110	VSSA	VSSA	DAC1 INT	DAC1 INT	DAC10 INT	DAC8 INT	DAC6 INT	DAC5 INT
0111	COMP1 INM1	COMP2 INM1	COMP3 INM1	COMP4 INM1	COMP5 INM1	COMP6 INM1	COMP7 INM1	COMP8 INM1
1000	COMP1 INM2	COMP2 INM2	COMP3 INM2	COMP4 INM2	COMP5 INM2	COMP6 INM2	COMP7 INM2	COMP8 INM2
1001	COMP2 INM	COMP1 INM	COMP4 INM	COMP3 INM	COMP6 INM	COMP5 INM	COMP8 INM	COMP7 INM

1. Other combinations are reserved.

Table 263. COMPx blanking source assignment (COMP1 to COMP8)

Banking source	BLANK SEL[3:0]	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
blk1	0001	tim1_oc5	tim1_oc5	tim1_oc5	tim1_oc5	tim1_oc5	tim1_oc5	tim1_oc5	tim1_oc5
blk2	0010	tim1_oc7	tim1_oc6	tim1_oc7	tim1_oc6	tim1_oc6	tim1_oc7	tim1_oc6	tim1_oc7
blk3	0011	tim2_oc3	tim2_oc3	tim2_oc4	tim9_oc1	tim2_oc3	tim2_oc4	tim9_oc1	tim9_oc1
blk4	0100	tim3_oc3	tim3_oc3	tim3_oc3	tim3_oc4	tim3_oc3	tim9_oc1	tim3_oc3	tim3_oc3
blk5	0101	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3	tim4_oc3
blk6	0110	tim8_oc5	tim8_oc5	tim8_oc5	tim8_oc5	tim8_oc5	tim8_oc5	tim8_oc5	tim8_oc5
blk7	0111	tim8_oc7	tim8_oc6	tim8_oc7	tim8_oc6	tim8_oc6	tim8_oc7	tim8_oc6	tim8_oc7
blk8	1000	tim12_oc1	tim12_oc1	tim12_oc1	tim12_oc1	tim12_oc1	tim12_oc1	tim12_oc1	tim12_oc1
blk9	1001	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5	tim20_oc5

Table 263. COMPx blanking source assignment (COMP1 to COMP8) (continued)

Banking source	BLANK SEL[3:0]	COMP1	COMP2	COMP3	COMP4	COMP5	COMP6	COMP7	COMP8
blk10	1010	tim20_oc7	tim20_oc6	tim20_oc7	tim20_oc6	tim20_oc6	tim20_oc7	tim20_oc6	tim20_oc7
blk11	1011	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1	tim15_oc1
blk12	1100	tim15_oc2	tim15_oc2	tim15_oc2	tim15_oc2	tim15_oc2	tim15_oc2	tim15_oc2	tim15_oc2
blk13	1101	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
blk14	1110	tim9_oc1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
blk15	1111	play1_out6	play1_out7	play1_out8	play1_out9	play1_out10	play1_out11	play1_out12	play1_out13

30.3.3 COMP reset and clocks

The COMP peripheral clock comp_pclk is provided by the clock controller is synchronous with the APB clock.

Warning: The polarity selection logic and the output redirection to the port work independently from the APB clock. This allows the comparator to work even in Stop mode. The interrupt line, connected to the NVIC of CPU, requires the APB clock (comp_pclk) to work. In absence of the APB clock, the interrupt signal comp_it cannot be generated.

30.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to ensure that the comparator programming cannot be altered in case of spurious register access or program counter corruption. For this purpose, the comparator control and status registers can be write-protected (read only). Once the programming is completed, the COMPx LOCK bit can be set. This causes the whole register to become read-only, including the COMPx LOCK bit. The write protection can only be removed by an MCU reset.

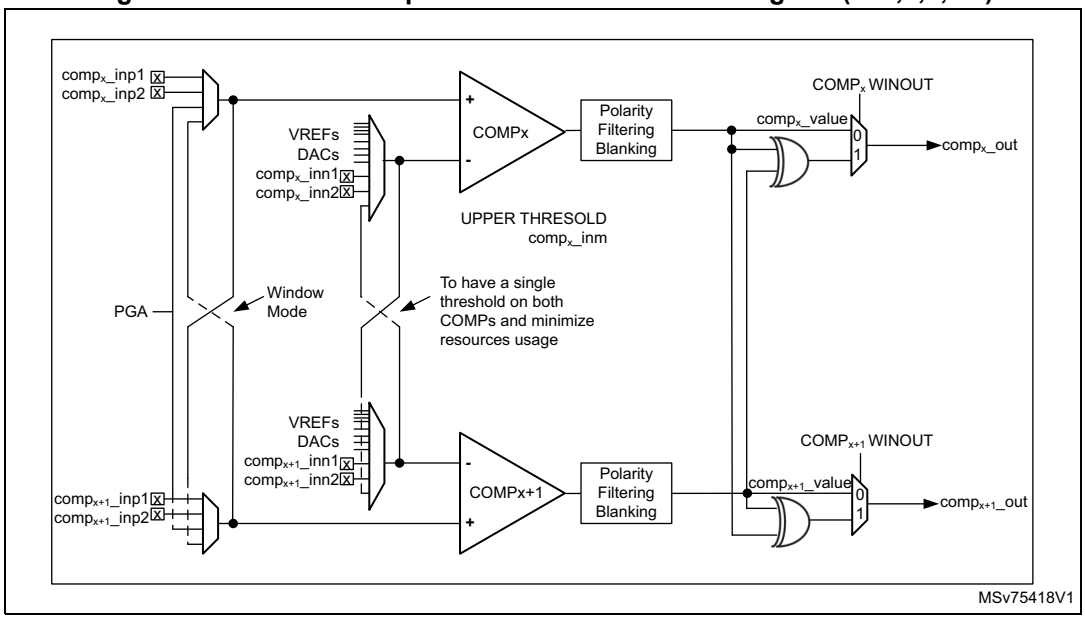
30.3.5 Window comparator

The purpose of the window comparator is to monitor the analog voltage if it is within the voltage range defined by the lower and upper threshold.

The two embedded comparators $COMP_x$ and $COMP_{x+1}$ with x=1,3,5,7... can be used to create a window comparator. The monitored analog voltage is connected to the non-inverting (plus) inputs of comparators connected together ($COMP_x$ or $COMP_{x+1}$). The upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators.

The output of the window comparator ($COMP_x$ or $COMP_{x+1}$) is selected with the control bit WINOUT.

Figure 281. Window comparators functional block diagram (x=1,3,5,7...)



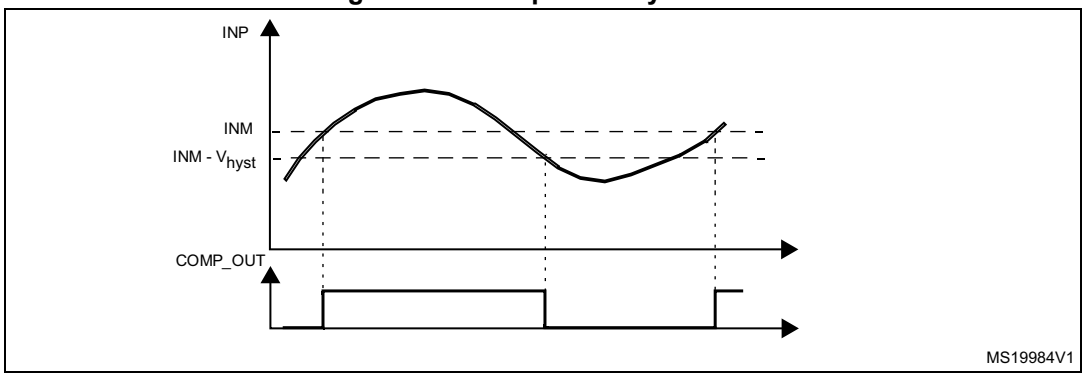
Important: when $COMP_x$ or $COMP_{x+1}$ have different secure mode, the window mode feature is disabled for both comparators with the following configuration (registers COMPx_CSR and COMPx+1_CSR unchanged) :

- WINOUT bit is forced internally to reset value
- In case INPSEL[1:0]=0x3, INPSEL[1:0] is forced internally to reset value
- In case INMSEL[3:0]=0x9, INMSEL[3:0] is forced internally to reset value

30.3.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

Figure 282. Comparator hysteresis



30.3.7 Deglitcher feature

In addition of the programmable hysteresis which is filtering a voltage amplitude, comparator high frequency noise on signal input can be filtered with the embedded deglitcher analog feature enabling the DGLT. The usage of the deglitcher is increasing the propagation delay of the comparator with few nanoseconds (refer to product datasheet).

High speed comparator are highly sensitive to noise. As a consequence, output comparator can oscillate for low input slope signal even if hysteresis is enabled. The deglitcher is an analog feature allowing to remove these comparator output toggling. The advantage is that the solution is purely analog : no need of extra digital circuitry and high speed clock, no extra latency due to digital filtering, negligible consumption. It is recommended to use the deglitcher coupled with hysteresis.

30.3.8 Digital signal filter

In addition of the programmable hysteresis and deglitcher features, comparator low frequency noise on input signal can be filtered with the digital signal filter. Based on a internal counter clocked by filter_clk (divided from comp_pclk): the output of the comparator is filtered considering if the logical level is stable for a programmable number of consecutive periods. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected). In the FLTR register, FSEL[3:0] register allows the selection of various filter_clk divider ratio and N value while FPSC[1:0] register controls the prescaler ratio from comp_pclk to filter_clk.

Figure 283. Comparator output digital filter (N=4)

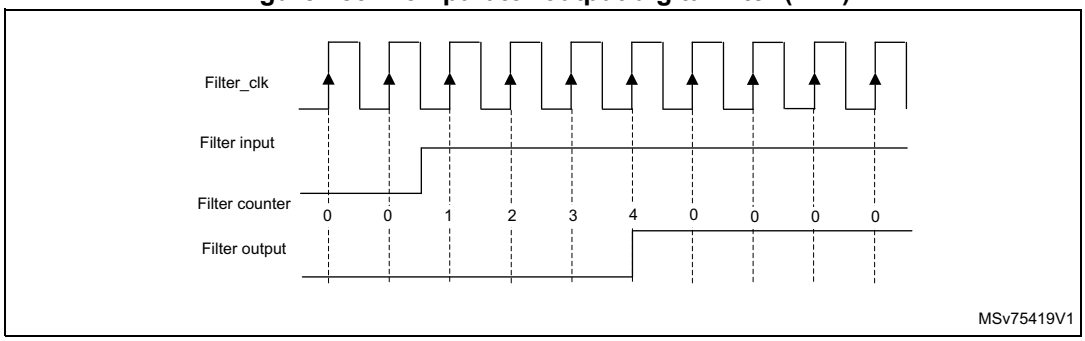
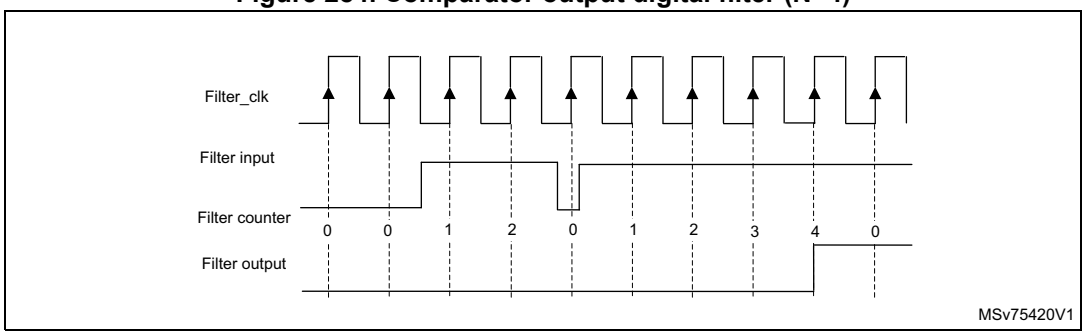


Figure 284. Comparator output digital filter (N=4)

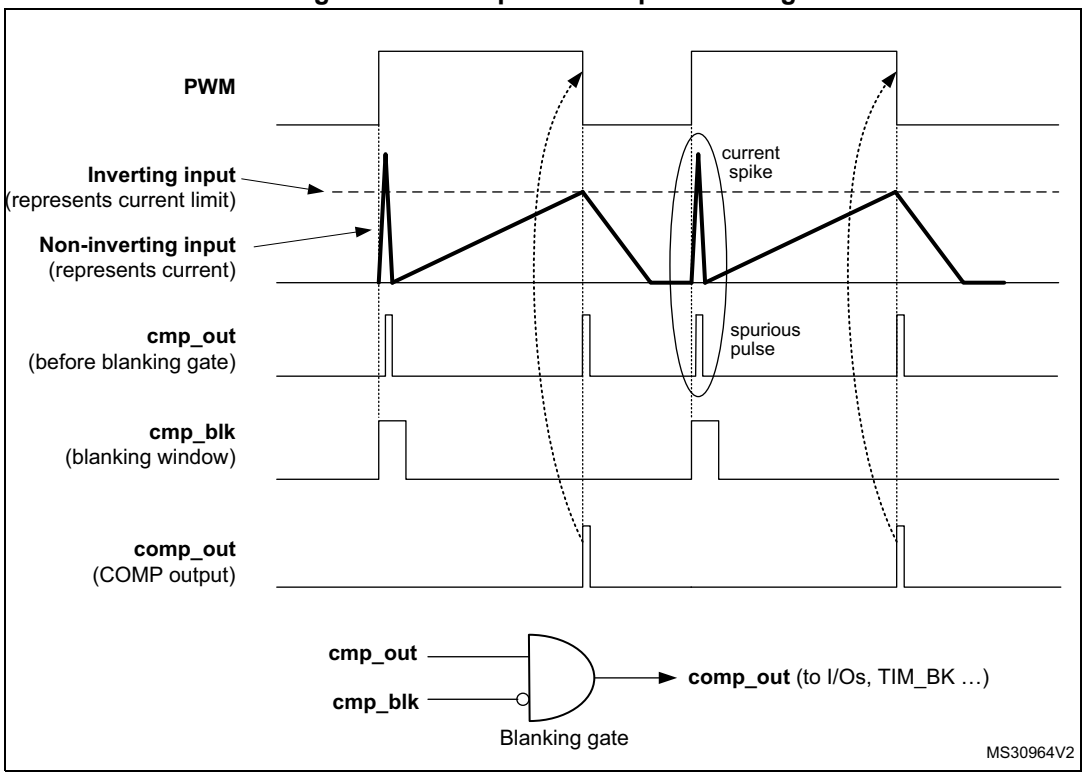


30.3.9 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It uses a blanking window defined with a timer output compare signal. Refer to the register description for selectable blanking signals.

The blanking signal gates the internal comparator output such as to clean the comp_out from spurious pulses due to current spikes, as depicted in *Figure 285* (the COMP channel number is not represented).

Figure 285. Comparator output blanking



30.3.10 COMP startup stabilization time

The comparator parameters and its output is undefined after the COMP enable (power-on) for a period defined as startup stabilization time, see the product datasheet for more details. If the comparator output is driving any critical stage of the design, it is recommended to redirect the comparator output to the device I/O through the GPIO alternate function or through IP configuration (such as Timers) after the stabilization time elapses.

30.4 COMP low-power modes

Table 264. Comparator behavior in low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Stop	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.

Note: The comparators cannot be used to exit the device from Sleep or Stop mode when the internal reference voltage is switched off.

30.5 COMP interrupts

The comparator outputs are internally connected to the extended interrupt and event controller (EXTI). Each comparator has its own EXTI line and can generate either interrupts or events to make the device exit low-power modes.

Interrupt through EXTI

Refer to interrupt and events section for more details.

Sequence to enable the COMP interrupt through EXTI block:

1. Enable the COMP and wait for stabilization.
2. Configure the EXTI line, receiving the compx_it signal, in interrupt mode, select the rising, falling or either-edge sensitivity and enable the EXTI line.
3. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines.

Table 265. Interrupt through EXTI control bits

Interrupt event	Event flag	Enable control bit	Exit Sleep mode	Exit Stop modes	Exit Standby mode
compx_it	Through EXTI	Through EXTI	Yes	Yes	N/A

30.6 COMP registers

30.6.1 Comparator x configuration register (COMPx_CSR)

Address offset: 0x00 + 0x400 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

The COMPx_CSR is the COMPx main configuration register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	DGLT	SCALEN	BRGEN	BLANKSEL[3:0]				HYST[2:0]		
rs	r					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL	WINOUT	Res.	Res.	Res.	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
rw	rw					rw	rw	rw	rw	rw	rw				rw

Bit 31 **LOCK:** Lock

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator x control register COMPx_CSR[31:0] and COMPx_FLTR[31:0]. When locked, all control bits and flags can be read only but not written. When unlocked, the control bits can also be written by software.

0: COMP_CxCR[31:0] and COMPx_FLTR[31:0] registers are read/write.

1: COMP_CxCR1[31:0] and COMPx_FLTR[31:0] are locked.

Bit 30 **VALUE**: COMPx output value

This bit is read-only. It reflects the level of the compx_value signal.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **DGLT**: DEGLITCHER enable

This bit, controlled by software, enables the operation of DEGLITCHER.

0: Disable

1: Enable

Bit 24 **SCALEN**: VREFINT scaler enable

This bit, controlled by software, enables the operation of VREFINT scaler at the inverting input of all comparator. To disable the VREFINT scaler, SCALEN bits of all COMPx_CSR registers must be reset. When the VREFINT scaler is disabled, the 1/4 VREFINT, 1/2 VREFINT, 3/4 VREFINT, and VREFINT inputs of the multiplexer must not be selected.

0: Disable

1: Enable

Bit 23 **BRGEN**: VREFINT scaler resistor bridge enable

This bit, controlled by software, enables the operation of resistor bridge in the VREFINT scaler. To disable the resistor bridge, BRGEN bits of all COMPx_CSR registers must be reset. When the resistor bridge is disabled, the 1/4 VREFINT, 1/2 VREFINT, and 3/4 VREFINT inputs of the input selector receive VREFINT voltage.

0: Disable

1: Enable

Bits 22:19 **BLANKSEL[3:0]**: Comparator x blanking signal select

This bitfield, controlled by software, selects the blanking signal for comparator channel x, as shown in *Table 263: COMPx blanking source assignment (COMP1 to COMP8)*.

Bits 18:16 **HYST[2:0]:** Comparator x hysteresis

This bitfield, controlled by software, selects the hysteresis of the comparator x.

000: No hysteresis

001: hysteresis level 1

010: hysteresis level 2

011: hysteresis level 3

100: hysteresis level 4

101: hysteresis level 5

110: hysteresis level 6

111: hysteresis level 7

Note: Refer to the datasheet for the real hysteresis level values.

Bit 15 **POL**: Comparator x polarity

This bit, controlled by software, selects the comparator x output polarity.

0: Non-inverted

1: Inverted

Bit 14 **WINOUT**: COMPx output selector

This bit, controlled by the software (if not locked), selects the COMPx window mode output.

0: $comp_x_out = comp_x_value$

1: $comp_x_out = WINOUT(comp_x)$

Note: WINOUT is forced to reset state when $COMP_x$ and $COMP_{x+1}$ secure mode differ.

Bits 13:10 Reserved, must be kept at reset value.

Bits 9:8 **INPSEL[1:0]**: COMPx channel non-inverting input selection

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input plus of COMPx channel.

Note: See Table 261: COMPx_INP non-inverting input assignment (COMP1 to COMP8) for more details. INPSEL[2:0] is forced to reset state when $COMP_x$ and $COMP_{x+1}$ secure mode differ.

Bits 7:4 **INMSEL[3:0]**: COMPx channel inverting input selection

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of COMPx channel.

Note: See Table 262: COMPx_INM inverting input assignment (COMP1 to COMP8) for more details. INMSEL[3:0] is forced to reset state when $COMP_x$ and $COMP_{x+1}$ secure mode differ.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN:** COMPx channel enable

This bit is set and cleared by software (only if LOCK not set). It enables the COMPx channel.

0: Disable

1: Enable

30.6.2 Comparator x filter configuration register (COMPx_FLTR)

Address offset: 0x04 + 0x400 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	FSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	FPSC[1:0]	
				rw	rw	rw	rw							rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **FSEL[3:0]:** COMPx digital filter configuration selection
Bits of this field are set and cleared by software (only if LOCK not set).They select the digital filter configuration of the COMPx.
0000: Filter disabled
0001: filter_sampling = comp_pclk, N = 2
0010: filter_sampling = comp_pclk, N = 4
0011: filter_sampling = comp_pclk, N = 8
0100: filter_sampling = filter_clk/2, N = 6
0101: filter_sampling = filter_clk/2, N = 8
0110: filter_sampling = filter_clk/4, N = 6
0111: filter_sampling = filter_clk/4, N = 8
1000: filter_sampling = filter_clk/8, N = 6
1001: filter_sampling = filter_clk/8, N = 8
1010: filter_sampling = filter_clk/16, N = 5
1011: filter_sampling = filter_clk/16, N = 6
1100: filter_sampling = filter_clk/16, N = 8
1101: filter_sampling = filter_clk/32, N = 5
1110: filter_sampling = filter_clk/32, N = 6
1111: filter_sampling = filter_clk/32, N = 8

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **FPSC[1:0]:** COMPx digital filter configuration selection
Bits of this field are set and cleared by software (only if LOCK not set). They select the digital filter clock configuration of the COMPx:
00: filter_clk = comp_pclk
01: filter_clk = comp_pclk/2
10: filter_clk = comp_pclk/4
11: filter_clk = comp_pclk/8

30.6.3 COMPx register map

Table 266. COMPx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 + 0x400 * (x - 1) (x=1 to 8)	COMPx_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	DGLT	SCALEN	BRGEN	BLANKSEL[3:0]				HYSTO[2:0]			POL	WINOUT	Res.	Res.	Res.	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0				0
0x04 + 0x400 * (x - 1) (x=1 to 8)	COMPx_FLTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	FPSC[2:0]	
	Reset value																					0	0	0	0							0	0

Refer to *Section 2.3.2* for the register boundary addresses.

31 Operational amplifier (OPAMP)

This section does not apply to the STM32G53x/54x devices as they do not include OPAMP.

31.1 Introduction

The devices embed three operational amplifier instances, OPAMP1, OPAMP2, and OPAMP3. OPAMPx used in the OPAMP section denotes any of these instances.

OPAMP contains an operational amplifier, an internal resistor divider, and multiplexers to set the OPAMP peripheral into a specific operating mode. In one of these, the operational amplifier inputs (INP and INM) and the output (OUT) are connected to external pins, which allows the use of OPAMP as a stand-alone operational amplifier device. The other OPAMP configurations allow the operation as a follower and as noninverting or inverting operational amplifier with programmable gain (with or without external filtering and/or biasing).

The programmable gain and the operating mode can be controlled through primary and secondary control register sets that can be activated in a time multiplex by the timer.

The operational amplifier output can be connected, through a multiplexer, to the OPAMP_VOUT pin or internally to an ADC or a COMP input channel.

The INP input of the OPAMP1 and OPAMP3 instances can be connected to an output of the internal DAC.

The operational amplifier allows normal and high speed (slew rate). Its input offset is compensated for through factory trimming that can be user modified if required.

As long as OPAMP is not required, it can be disabled to save power.

31.2 OPAMP main features

- Rail-to-rail input voltage range
- Low input bias current
- Low input offset voltage (factory compensated for)
- High frequency bandwidth
- Normal and high speed (slew rate)
- Timer-controlled operation

Note: Refer to the product datasheet for detailed OPAMP characteristics.

31.3 OPAMP implementation

The device has three OPAMP instances, OPAMP1, OPAMP2, and OPAMP3. *Table 267* shows the pin and internal signal per-instance specific naming and provides the assignment to I/Os and the possible connection of the internal signals.

The *Generic name* column provides the generic pin and internal signal names as per *Figure 286: OPAMP block diagram*. The *Instance specific name* column provides specific pin and internal signal names pertaining to each instance. The *Assignment or possible*

connection column provides the assignment to I/O pins for signals on pins (in upper-case letters) and the possible connections of internal signals (in lower-case letters).

Table 267. OPAMP pin and signal assignment per instance

Generic name	Instance-specific name	Assignment or possible connection	Comment
OPAMP1			
OPAMP_VINM0	OPAMP1_VINM0	PA3	Selected as additional I/O function.
OPAMP_VINM1	OPAMP1_VINM1	PC5	
OPAMP_VINP0	OPAMP1_VINP0	PA1	
OPAMP_VINP1	OPAMP1_VINP1	PA3	
OPAMP_VINP2	OPAMP1_VINP2	PA7	
opamp_dac_in	opamp1_dac_in	dac5_ch1	-
OPAMP_VOUT	OPAMP1_VOUT	PA2	Selected as additional I/O function.
opamp_out	opamp1_out	adc1_in13 adc3_in19 comp1_inp2 comp2_inp2	Selected in the ADC/COMP.
opamp_pclk	opamp1_pclk3	-	Enabled/disabled through RCC.
OPAMP2			
OPAMP_VINM0	OPAMP2_VINM0	PA5	Selected as additional I/O function.
OPAMP_VINM1	OPAMP2_VINM1	PC5	
OPAMP_VINP0	OPAMP2_VINP0	PA7	
OPAMP_VINP1	OPAMP2_VINP1	PB14	
OPAMP_VINP2	OPAMP2_VINP2	PB0	
OPAMP_VINP3	OPAMP2_VINP3	PC1	
OPAMP_VOUT	OPAMP2_VOUT	PA6	
opamp_out	opamp2_out	adc1_in19 adc2_in16 comp3_inp2 comp4_inp2	Selected in the ADC/COMP.
opamp_pclk	opamp2_pclk3	-	Enabled/disabled through RCC.
OPAMP3			
OPAMP_VINM0	OPAMP3_VINM0	PB2	Selected as additional I/O function.
OPAMP_VINM1	OPAMP3_VINM1	PB10	
OPAMP_VINP0	OPAMP3_VINP0	PB0	
OPAMP_VINP1	OPAMP3_VINP1	PB13	
OPAMP_VINP2	OPAMP3_VINP2	PA1	
opamp_dac_in	opamp3_dac_in	dac6_ch1	-
OPAMP_VOUT	OPAMP3_VOUT	PB1	Selected as additional I/O function.

Table 267. OPAMP pin and signal assignment per instance (continued)

Generic name	Instance-specific name	Assignment or possible connection	Comment
opamp_out	opamp3_out	adc2_in19 adc3_in13 comp5_inp2 comp6_inp2	Selected in the ADC/COMP.
opamp_pclk	opamp3_pclk3	-	Enabled/disabled through RCC.

31.4 OPAMP functional description

Each OPAMP instance can individually be enabled. When disabled, its output is in a high-impedance state. When enabled, it can either operate in functional mode or in calibration mode.

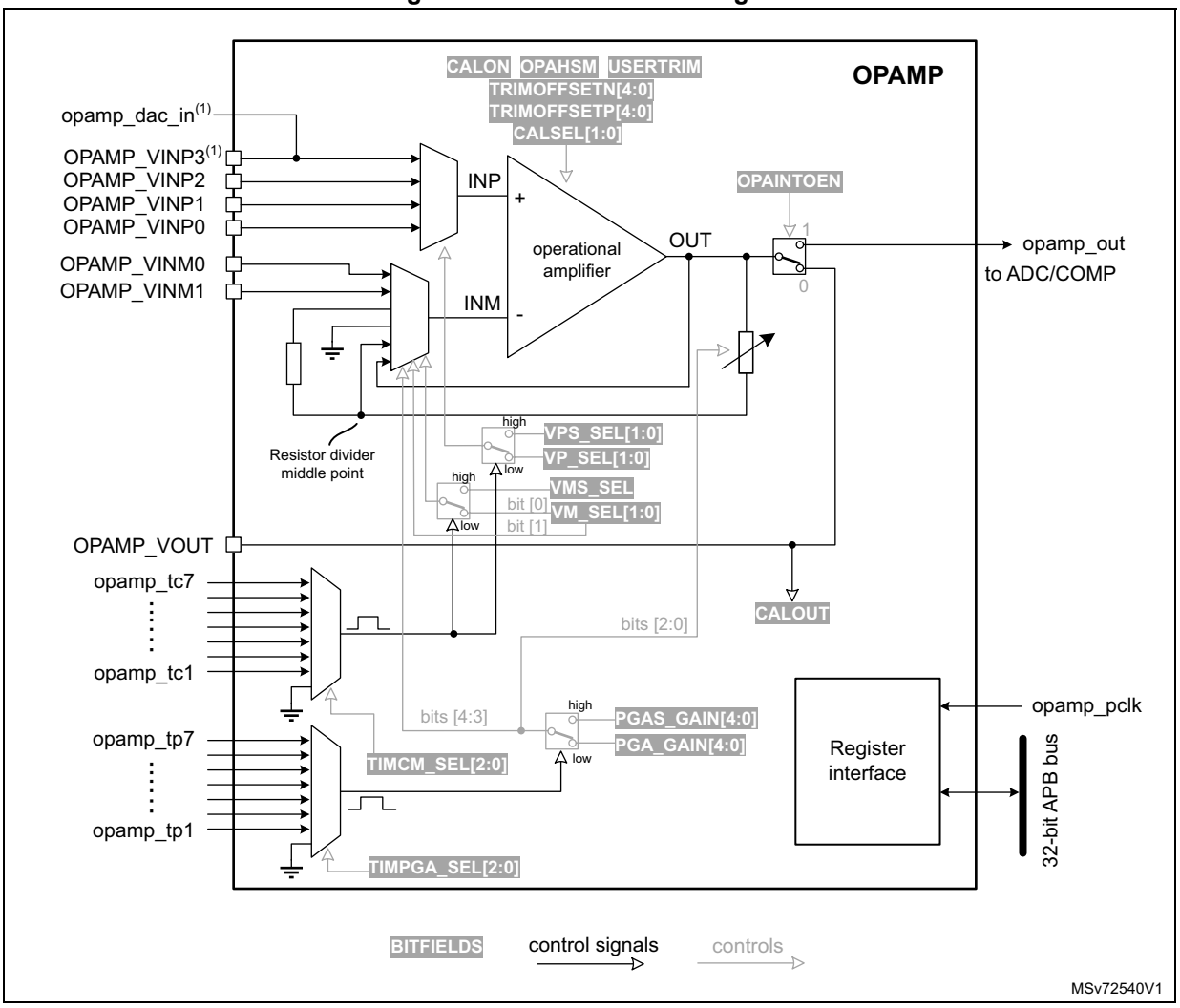
In functional mode, the inputs and the output of the operational amplifier are connected as described in *Section 31.4.8*. In calibration mode, they are all disconnected.

Caution: The GPIOs connected to the operational amplifier terminals through the OPAMP multiplexers must be configured in analog mode (default state), through the corresponding GPIOx_MODER register.

31.4.1 OPAMP block diagram

The following figure shows the block diagram of the OPAMP peripheral. The calibration multiplexers and some control bitfields are not shown.

Figure 286. OPAMP block diagram



1. Either OPAMP_VINP3 pin or DAC output, depending on the OPAMP instance. Refer to *Section 31.3: OPAMP implementation*.

31.4.2 OPAMP pins and internal signals

The following table shows the pins used by the OPAMP peripheral. The pin names correspond to their configuration for use as OPAMP inputs or outputs. Refer to *Section 31.3: OPAMP implementation* for assignment of each to an I/O.

Table 268. OPAMP pins

Pin name	Pin type	Description
OPAMP_VINP0 OPAMP_VINP1 OPAMP_VINP2 OPAMP_VINP3	I	Selectable as INP noninverting input of the operational amplifier.

Table 268. OPAMP pins (continued)

Pin name	Pin type	Description
OPAMP_VINM0 OPAMP_VINM1	I	Selectable as INM inverting input of the operational amplifier.
OPAMP_VOUT	O	Operational amplifier output. Usable for other purposes when the operational amplifier output OUT is redirected to an internal ADC/COMP channel.

Caution: The input and output pins connected, through the OPAMP multiplexers, to the operational amplifier terminals INP, INM, and OUT must be configured in analog mode (default state), through the corresponding GPIOx_MODER register.

The following table lists the OPAMP internal signals. Refer to *Section 31.3: OPAMP implementation* for possible connection of each to internal resources and for specialization of the signal name for each OPAMP instance.

Table 269. OPAMP internal input/output signals

Pin name	Pin type	Description
opamp_tc1 opamp_tc2 opamp_tc3 opamp_tc4 opamp_tc5 opamp_tc6 opamp_tc7	I	Signal from a timer channel, selectable as control signal switching between primary and secondary controls for the input multiplexers.
opamp_tp1 opamp_tp2 opamp_tp3 opamp_tp4 opamp_tp5 opamp_tp6 opamp_tp7	I	Signal from a timer channel, selectable as control signal switching between primary and secondary controls for the programmable gain and for setting the PGA flavor.
opamp_dac_in	I	Output of a DAC channel, selectable as noninverting operational amplifier input. Exclusive with the OPAMP_VINP3 pin.
opamp_pclk	I	Clock for OPAMP registers.
opamp_out	O	Operational amplifier output to an internal ADC/COMP channel, when redirected through the OPAINTOEN bit.

31.4.3 OPAMP reset and clocks

The OPAMP clock opamp_pclk provided by the clock controller is generated from the PCLK3 (APB3 clock). To access the OPAMP control registers, it is necessary to set the corresponding OPAMPxEN bit of the RCC_APB3ENR register.

The OPAMP registers can be reset through the corresponding OPAMPxRST bit of the RCC_APB3RSTR register.

31.4.4 OPAMP output redirection to an internal ADC/COMP channel

The operational amplifier output OUT can be redirected internally to an ADC or a COMP channel, by setting the OPAINTOEN bit of the OPAMP_CSR register. In this case, the I/O pin OPAMP_VOUT on which the operation amplifier output OUT is mapped can be used for another purpose. See *Table 267* for possible ADC or COMP input channel assignment to OPAMP instances.

When OUT is connected to an ADC/COMP input channel, it is not connected to the OPAMP_VOUT pin. When the output on the OPAMP_VOUT and the connection to an ADC channel are required simultaneously, the output OUT must be routed to the OPAMP_VOUT pin and the ADC input selected from the OPAMP_VOUT pin.

31.4.5 OPAMP initial configuration

By default (upon reset):

- The operational amplifier inputs INP and INM and output OUT are connected to I/O pins.
- The registers for offset compensation contain factory trimming values and are in read-only access mode.
- The speed (slew rate) is set to normal.
- The timer-controlled operation of multiplexers is disabled.
- OPAMP is disabled.

31.4.6 OPAMP operational amplifier speed

The initial operational amplifier speed (slew rate, response) is normal. Setting the OPAHSM bit of the OPAMP_CSR register increases the speed. Both normal and high-speed mode characteristics are defined in the datasheet.

31.4.7 OPAMP enable

When disabled (default upon reset), the OPAMP control and calibration registers can still be modified. The operational amplifier output OUT is in high impedance state.

It is recommended to update the OPAMP registers while the OPAMP peripheral is disabled, to avoid spurious effects on the output.

OPAMP is enabled by setting the OPAEN bit of the OPAMP_CSR register. The INP and INM inputs and the OUT output are connected through the embedded multiplexers according to the control register settings and the timer control signals.

31.4.8 OPAMP operating modes

OPAMP can be configured to different operating modes or use cases, through:

- OPAMP control registers
- GPIOx_MODER registers, configuring the I/Os selected as OPAMP inputs and output
- ADC registers, selecting an ADC input channel when the operational amplifier output is redirected to the ADC
- COMP registers, selecting a COMP input channel when the operational amplifier output is redirected to the COMP
- External components, defining the gain, bias, or filtering parameters

The principal operating modes are:

- A stand-alone operational amplifier
- A follower
- A programmable-gain amplifier (PGA)

Refer to the following subsections for details and for different PGA flavors.

Through the OPAINTOEN bit of the *OPAMP control/status register (OPAMP_CSR)*, the operational amplifier output OUT can either be connected to the I/O pin selected as OPAMP output (default) or to an internal ADC/COMP input channel. However, the latter option cannot be used when the operational amplifier output on the pin is required.

The OPAHSM bit (at zero by default) allows increasing the operational amplifier speed (slew rate).

The USERTRIM bit can enable writing to the input offset trimming register, to allow modifying the by-default factory trimming values by the user.

The impedance of the input signal must be low enough to prevent the OPAMP input current from creating a voltage drop in the source, thus altering the input voltage. Refer to the electrical characteristics section in the datasheet for further details.

Stand-alone operational amplifier

For this use, ensure the following settings.

- Select a pin for the INP input (as opposed to a DAC output), with VP_SEL[2:0] and VPS_SEL[2:0] bitfields.
- Set the VM_SEL[1:0] bitfield to 0X, to ensure that a pin is selected for the INM input.
- Clear OPAINTOEN bit, to direct the operational amplifier output to the OPAMP_VOUT pin.

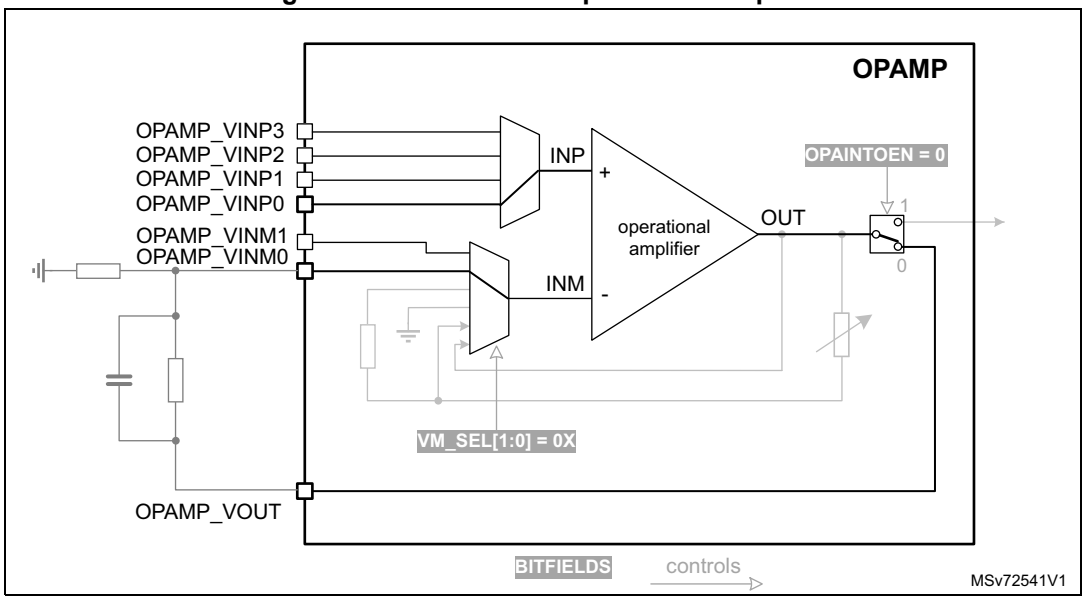
With these settings, the INP, INM inputs and the OUT output of the operational amplifier are connected to pins and to no other internal structure.

Note: The OPAMP_CSR, OPAMP_TCMR, and GPIOx_MODER registers in their default state configure OPAMP as stand-alone operational amplifier, with normal speed, factory offset trimming values, and with OPAMP_VINP0 and OPAMP_VINM0 selected for INP and INM, respectively. The OPAMP is disabled so the OUT and OPAMP_OUT are in high-Z state.

Once enabled, OPAMP operates as a stand-alone operational amplifier, with the gain, filtering, and biasing determined by external routing and components.

For this use, the operational amplifier output OUT must be directed to the OPAMP_VOUT pin and not to the internal ADC/COMP channel.

Figure 287. Stand-alone operational amplifier



Follower

To use OPAMP as a follower with an internal feedback, set the following bitfields as indicated:

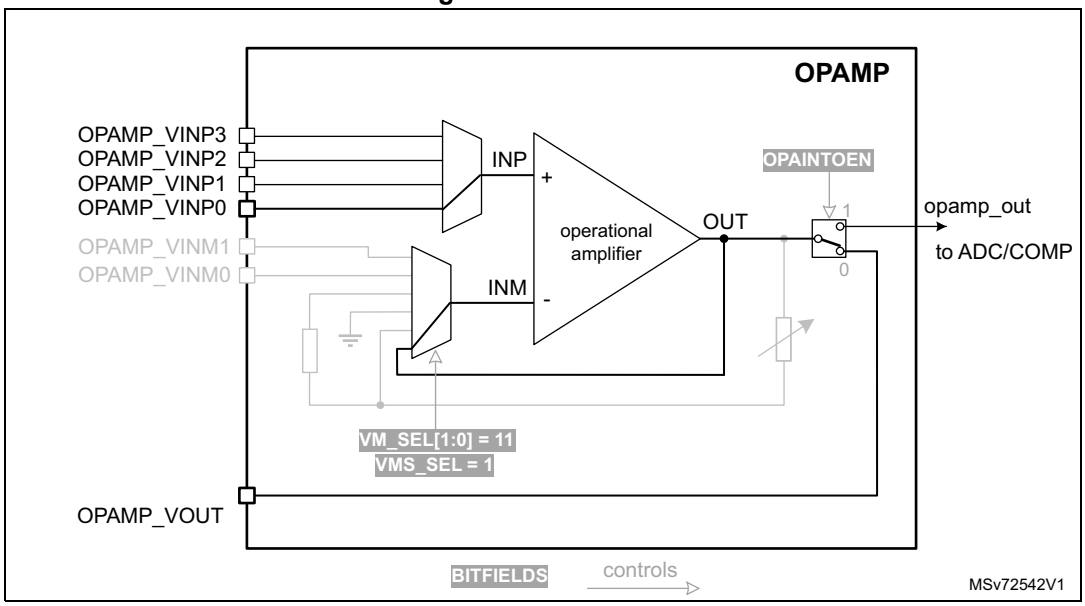
- VM_SEL[1:0] to 11 and VMS_SEL to 1, to loop the operational amplifier output back to its inverting input INM
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select for the noninverting input INP

Once enabled, the operational amplifier buffers the voltage selected on its noninverting input INP.

Note: The GPIOs corresponding to OPAMP_VINMy are free for another use.

In this mode, OPAMP only takes two pins if a pin is selected for INP noninverting input. It only takes a single pin if an internal DAC output is selected for INP noninverting input (on instances supporting this). Refer to Figure 31.3: OPAMP implementation.

Figure 288. Follower



Programmable-gain amplifier

When OPAMP is configured as a a programmable-gain amplifier (PGA), the internal resistor divider with a programmable resistor value determines the amplifier gain. Refer to the datasheet for the values of the internal resistors.

The following sections describe different PGA flavors.

Noninverting PGA

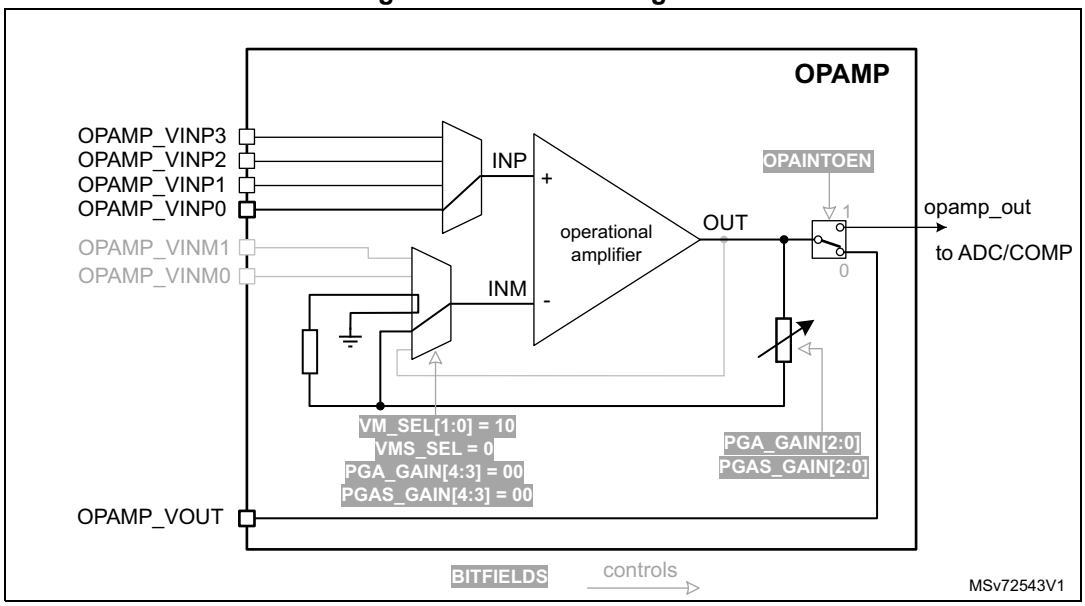
To use OPAMP as a noninverting PGA without filtering, set the following bitfields as indicated:

- VM_SEL[1:0] to 10 and VMS_SEL to 0, to connect the resistor divider middle point to the inverting operational amplifier input
- PGA_GAIN[4:0] and PGAS_GAIN[4:0] to 00XXX, with XXX according to the desired positive gain
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select as noninverting operational amplifier input

Once enabled, OPAMP outputs a voltage that corresponds to the selected input pin voltage multiplied by the selected gain value.

Note: To avoid saturation, keep the product of the input voltage and the selected gain value below V_{DDA}.

Figure 289. Noninverting PGA



Noninverting PGA with filtering

To use OPAMP as a noninverting PGA with filtering, set the following bitfields as indicated:

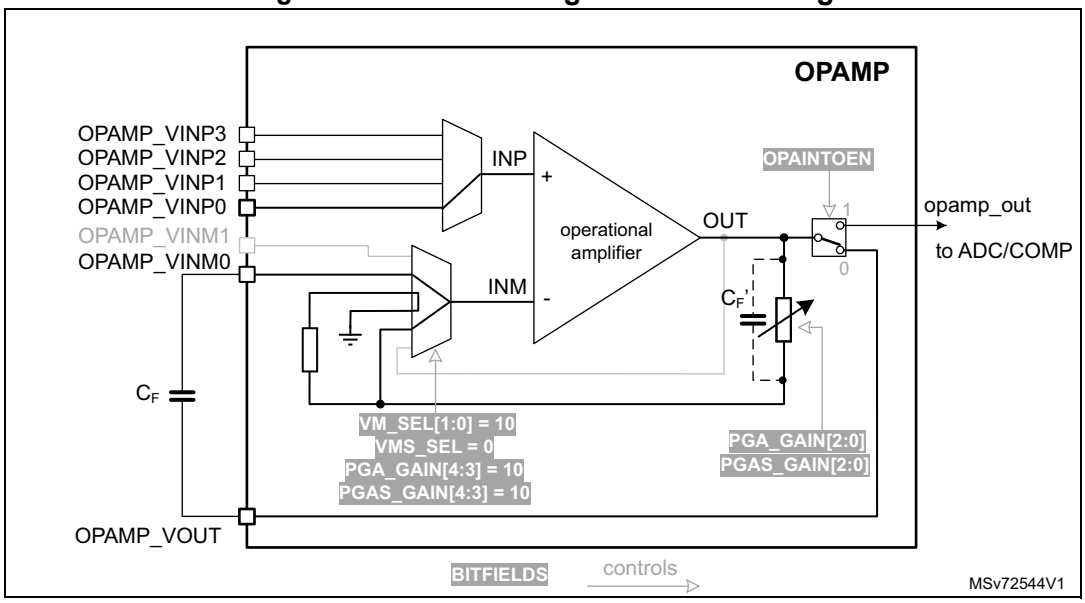
- VM_SEL[1:0] to 10 and VMS_SEL to 0, to connect the resistor divider middle point to the inverting operational amplifier input
- PGA_GAIN[4:0] and PGAS_GAIN[4:0] to 10XXX, with XXX according to the desired positive gain
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select as noninverting operational amplifier input

Once enabled, OPAMP outputs a voltage that corresponds to the selected input pin voltage multiplied with the selected gain (for static signals).

Any external components connected between the OPAMP_VOUT pin and the OPAMP_VINM0 pin are put in parallel with the internal programmable resistor. For example, connecting a capacitor C_F as shown in the following figure has a filtering effect equivalent to C_F' of the same capacitance. The gain for dynamic signals then also depends on the signal frequency and the cut-off frequency of the filter.

For this use, the operational amplifier output OUT must be directed to the OPAMP_VOUT pin and not to the internal ADC/COMP channel.

Figure 290. Noninverting PGA with filtering



Noninverting PGA with external bias, or inverting PGA

To use OPAMP as a noninverting PGA with external bias or as an inverting PGA, set the following bitfields as indicated:

- VM_SEL[1:0] to 10 and VMS_SEL to 0, to connect the resistor divider middle point to the inverting operational amplifier input
- PGA_GAIN[4:0] and PGAS_GAIN[4:0] to 01XXX, with XXX according to the desired positive gain
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select as noninverting operational amplifier input

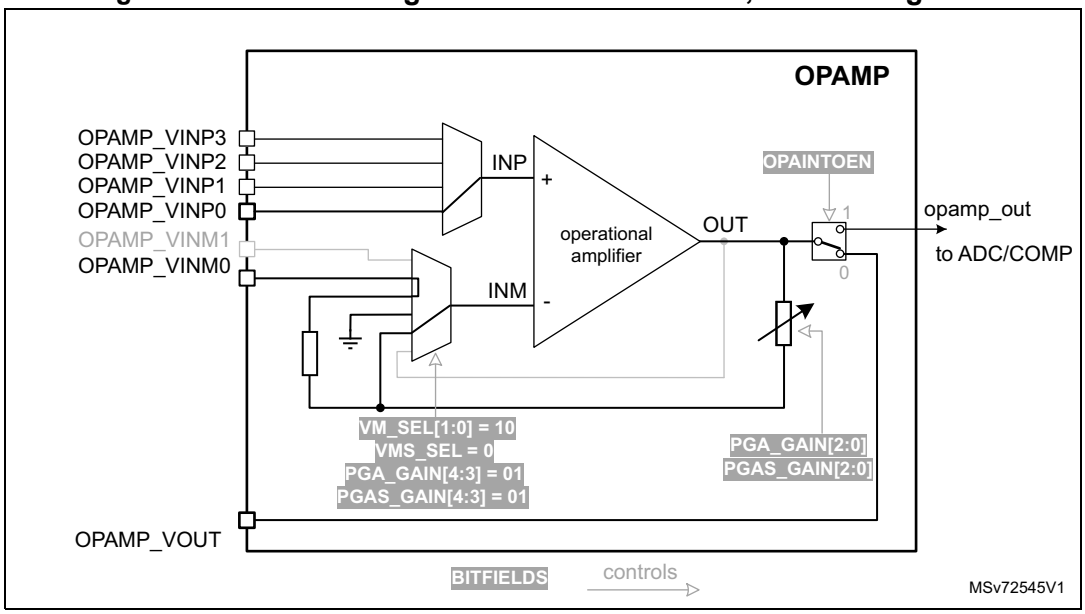
For the signal on the selected OPAMP_VINPy pin, OPAMP operates as a noninverting amplifier with the selected positive gain. The OPAMP_VINM0 pin is used as a reference (bias).

For the signal on the OPAMP_VINM0 pin, OPAMP operates as an inverting amplifier with the selected negative gain. The selected OPAMP_VINPy pin is used as a reference.

Note: The final gain depends on the output impedance of source connected to the OPAMP_VINM0 pin.

For this use, the operational amplifier output OUT must be directed to the OPAMP_VOUT pin and not to the internal ADC/COMP channel.

Figure 291. Noninverting PGA with external bias, or inverting PGA



Inverting or noninverting PGA with external bias or/and filtering

To use OPAMP as a noninverting PGA with external bias, or as an inverting PGA with filtering, set the following bitfields as indicated:

- VM_SEL[1:0] to 10 and VMS_SEL to 0, to connect the resistor divider middle point to the inverting operational amplifier input
- PGA_GAIN[4:0] and PGAS_GAIN[4:0] to 11XXX, with XXX according to the desired positive gain
- VP_SEL[1:0] and VPS_SEL[1:0] according to the pin or internal DAC output to select as noninverting operational amplifier input

For the signal on the selected OPAMP_VINPy pin, OPAMP operates as a noninverting amplifier with the selected positive gain. The voltage on the OPAMP_VINM0 pin is used as a reference (bias).

For the signal on the OPAMP_VINM0 pin, OPAMP operates as an inverting amplifier with the selected negative gain. The voltage on the selected OPAMP_VINPy pin is used as a reference (bias). The output impedance of the biasing voltage source impacts the total gain.

In both cases, any external components connected between the OPAMP_VOUT pin and the OPAMP_VINM1 pin are put in parallel with the internal programmable resistor. For example, connecting a capacitor C_F as shown in the following figure has a filtering effect equivalent to C_F' of the same capacitance. The gain for dynamic signals then also depends on the signal frequency and the cut-off frequency of the filter.

For this use, the operational amplifier output OUT must be directed to the OPAMP_VOUT pin and not to the internal ADC/COMP channel.

Figure 292. Inverting or noninverting PGA with external bias or/and filtering

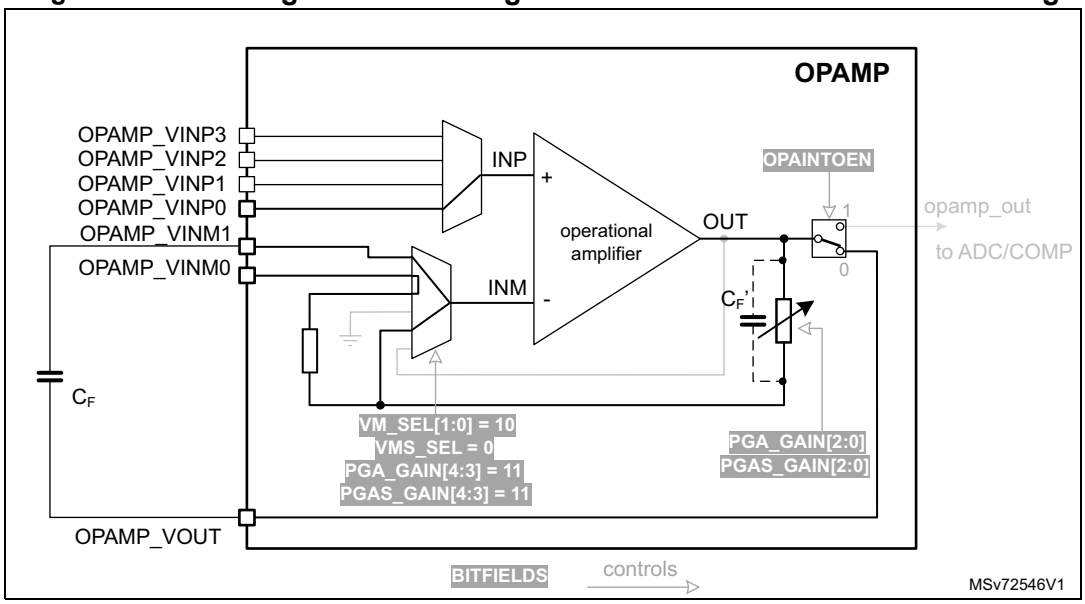
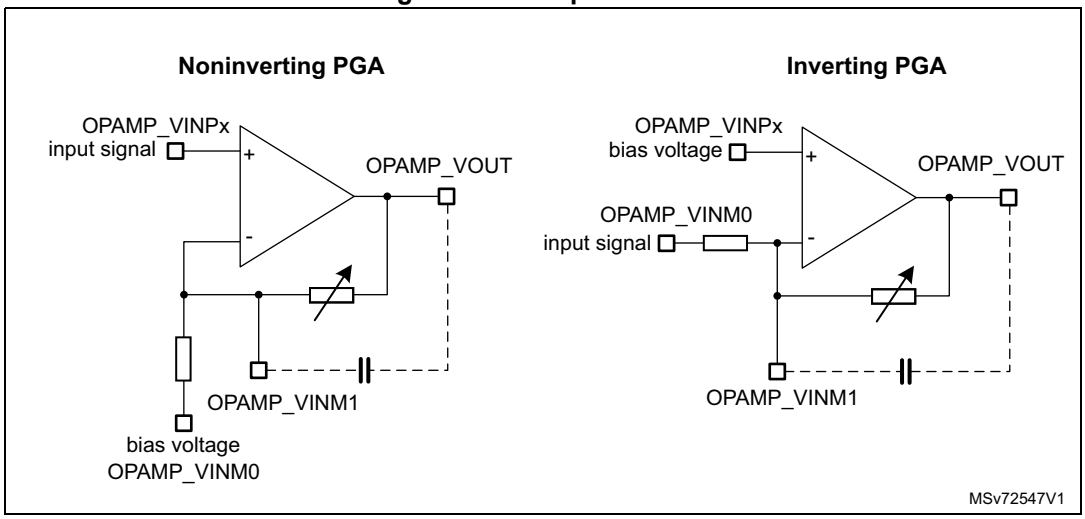


Figure 293. Simplified view



31.4.9 OPAMP programmable gain

In PGA mode, OPAMP allows selecting one of discrete positive (noninverting configuration) or negative (inverting configuration) gain values. The selection is done through the bits [2:0] of the PGA_GAIN[4:0] bitfield of the OPAMP_CSR register or through the bits [2:0] of the PGAS_GAIN[4:0] bitfield of the OPAMP_TCMR register. The control bitfield is selected with the control signal at the output of the opamp_tpX selector.

In noninverting mode, the programmable gain solely depends on the internal resistive divider. In inverting mode, it not only depends on the internal resistors but also on the output impedance of the signal source. The higher it is, the greater impact it has to the final gain.

31.4.10 OPAMP operational amplifier offset compensation

The peripheral incorporates a trimming circuitry that impacts the operational amplifier input offset. The trimming circuitry is controlled through the TRIMOFFSETN[4:0] (for the N differential pair) and TRIMOFFSETP[4:0] (for the P differential pair) bitfields of the *OPAMP control/status register (OPAMP_CSR)*.

The TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] bitfields keep the trimming value for the N differential pair and P differential pair, respectively.

Upon reset, these registers are loaded with factory trimming values stored in the nonvolatile memory. As the USERTRIM bit is low, the TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] control bitfields are read-only.

Note: The factory trimming values result from a calibration procedure run on the production line at specific production conditions.

When the offset compensation obtained with the factory trimming values is not acceptable for a specific use case (supply voltage, operating temperature), the user software can apply different trimming values.

To modify the trimming values, the user software has first to set the USERTRIM bit to allow writing into the TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] control bitfields.

To retain the user trimming values, the user software has to write them into a nonvolatile memory. It typically retrieves them from the non-volatile memory and writes into TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] at the startup, before the first use of OPAMP.

The resolution of the offset trimming allows reducing the input offset voltage to a value defined in the datasheet, at a defined supply voltage and die temperature.

31.4.11 OPAMP calibration

The calibration allows finding the trimming values that minimize the operational amplifier offset.

Setting the CALON bit isolates the operational amplifier INP and INM inputs and connects them to the internal reference voltage level selected through the CALSEL[1:0] bitfield. The operational amplifier is in a comparator (high gain, open loop) mode in which its output transits from one rail to another when the voltage level on one input crosses the voltage level on the other input.

The CALOUT flag reflects this digital value (lower rail or upper rail) of the OPAMP_VOUT pin.

The CALSEL[1:0] bitfield values 01 (low reference voltage) and 11 (high reference voltage) are intended for use in the calibration procedure. The value 01 is for trimming the PMOS through the TRIMOFFSETP[4:0] bitfield, the value 11 for trimming the NMOS through the TRIMOFFSETN[4:0] bitfield.

The calibration is carried out by software. Starting from the trimming bitfield at 0x1 and the CALOUT flag at one, the procedure consists in incrementing the trimming value until the CALOUT flag toggles to zero. The last trimming value is the one that compensates for the operational amplifier offset of the given OPAMP instance, and for the given supply voltage, die temperature, and differential pair stage (PMOS or NMOS). Upon each trimming value incrementation, the software must wait for at least the $t_{OFFTRIM}$max period defined in the product datasheet, before reading the CALOUT flag and potentially proceeding to the next incrementation step.

Note: The closer is the trimming value to the offset compensation point, the longer takes the output voltage to respond (without ever exceeding $t_{OFFTRIM}max$).

The full calibration consists of four calibration loops as listed in the following table. They can be executed in any order. The CALON bit must be set for executing any calibration loop. The calibration loops for the operational amplifier speed (slew rate) that is not required in the application can be omitted.

Table 270. Calibration loops

Calibration loop	Control bitfield		
	CALSEL[1:0]	TRIMOFFSETN[4:0]	TRIMOFFSETP[4:0]
NMOS	11	increment	keep as is
PMOS	01	keep as is	increment

Different OPAMP instances can be calibrated at any time, even at the same time.

Calibration initialization

To initialize the offset trimming value calibration:

- Enable OPAMP, by setting OPAEN.
- Make the trimming bitfields writable, by setting USERTRIM.
- Write TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] bitfields with 0b00001.

Calibration loop sequence

To execute a calibration loop, go through the following steps:

1. Select the calibration loop, with the CALSEL[1:0] bitfield. That is, the speed (slew rate) and the differential pair end (NMOS or PMOS) to calibrate.
2. Increment the appropriate trimming bitfield (TRIMOFFSETN[4:0] for NMOS, TRIMOFFSETP[4:0] for PMOS), keeping the other trimming bitfield as is.
3. Wait for at least the $t_{OFFTRIMmax}$ period.
4. Check the CALOUT flag (OPAINTOEN = 0) or the ADC output (OPAINTOEN = 1). If high, go to step 2 (loop back).
5. The trimming bitfield is calibrated. Store the calibrated trimming value if required.

For a given speed (slew rate), execute both the NMOS and the PMOS calibration loops.

Note: During the calibration loop execution, the external connection of the operational amplifier output must not pull up or down currents exceeding 500 µA.

Calibrating when operational amplifier output goes to an ADC/COMP input

With the operational amplifier output redirected to an internal ADC/COMP input channel (OPAINTOEN bit set), the CALOUT flag does not reflect the operational amplifier output. There are two options to address this.

The first option is to use, during the calibration, the ADC output as indicator of reaching the target trimming value. An ADC output value close to the maximum is equivalent to CALOUT at 1. An ADC output value close to zero is equivalent to CALOUT at zero.

The second option is to clear the OPAINTOEN bit for the time of calibration to make the CALOUT flag operate. When the calibration is completed, set OPAINTOEN to redirect the

operational amplifier output back to an ADC/COMP input channel. As during the calibration the internal signal sent to the OPAMP_VOUT pin toggles, care must be taken to prevent any conflict on the pin. For example, use the pin in the application as an input.

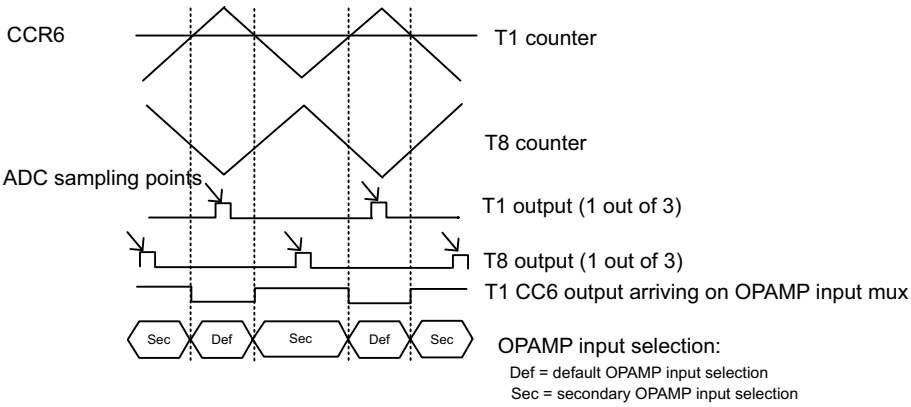
31.4.12 Timer-controlled operation

Certain timer channels (see *Table 271*) can toggle the control of the operational amplifier multiplexers between the primary configuration defined through the OPAMP_CSR register and the secondary configuration defined through the OPAMP_TCMR register.

Once configured, the mechanism operates even when the OPAMP clock is disabled.

As an example, this can be used for dual motor control. The following figure illustrates the use of a timer compare output toggling the OPAMP primary (default) and secondary configurations, to measure alternatively the currents of the three phases of one motor then of the other motor.

Figure 294. Timer-controlled operation for a dual motor application



Timer-controlled input selection

This operation is activated by writing a value other than zero to the TIMCM_SEL[2:0] bitfield of the OPAMP_TCMR register, thus selecting one of opamp_tcX signals. As long as the selected signal is low, the input selectors are controlled with the VP_SEL[1:0] and VM_SEL[1:0] bitfields of the OPAMP_CSR register. As long as it is high, the input selectors are controlled with the VPS_SEL[1:0] and VMS_SEL bitfields of the OPAMP_TCMR register. Writing zero to TIMCM_SEL[2:0] (default upon reset) makes the input selectors steadily controlled with VP_SEL[1:0] and VM_SEL[1:0].

The following table gives the correspondence of the opamp_tcX signals to timer signals.

Table 271. Timer-controlled input selection

TIMCM_SEL[2:0]	Signal name	Timer connection
000	-	None
001	opamp_tc1	tim1_oc6
010	opamp_tc2	tim2_oc4
011	opamp_tc3	tim8_oc6

Table 271. Timer-controlled input selection (continued)

TIMCM_SEL[2:0]	Signal name	Timer connection
100	opamp_tc4	tim12_oc1
101	opamp_tc5	tim15_oc2
110	opamp_tc6	tim20_oc6
111	opamp_tc7	Reserved

Timer-controlled programmable gain and PGA flavor

This operation is activated by writing a value other than zero to the TIMPGA_SEL[2:0] bitfield of the OPAMP_TCMR register, thus selecting one of opamp_tpX signals. As long as the selected signal is low, the programmable gain and PGA flavor selectors are controlled with the PGA_GAIN[4:0] bitfield of the OPAMP_CSR register. As long as it is high, the input selectors are controlled with the PGAS_GAIN[4:0] bitfield of the OPAMP_TCMR register. Writing zero to TIMPGA_SEL[2:0] (default upon reset) makes the programmable gain and PGA flavor selectors steadily controlled with PGA_GAIN[4:0].

The following table gives the correspondence of the opamp_tpX signals to timer signals.

Table 272. Timer-controlled programmable gain selection

TIMPGA_SEL[2:0]	Signal name	Timer connection
000	-	None
001	opamp_tp1	tim1_oc6
010	opamp_tp2	tim3_oc2
011	opamp_tp3	tim8_oc6
100	opamp_tp4	tim9_oc1
101	opamp_tp5	tim15_oc2
110	opamp_tp6	tim20_oc6
111	opamp_tp7	Reserved

31.5 OPAMP in low-power modes

Table 273. Effect of low-power modes on OPAMP

Mode	Description
Sleep	No effect.
Stop	No effect, OPAMP register contents are kept.
Standby	The OPAMP registers are powered down and must be re-initialized after exiting Standby mode.

31.6 OPAMP registers

31.6.1 OPAMP control/status register (OPAMP_CSR)

Address offset: 0x00

Reset value: 0xXXXX 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	CAL OUT	Res.	TRIMOFFSETN[4:0]					TRIMOFFSETP[4:0]					PGA_GAIN[4:2]		
rw	r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGA_GAIN[1:0]		CALSEL[1:0]		CALON	Res.	Res.	OPA INTOEN	OPA HSM	VM_SEL[1:0]		USER TRIM	VP_SEL[1:0]		FORCE _VP	OPAEN
rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP_CSR register lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.
This bit is used to configure the OPAMP_CSR register as read-only.
0: OPAMP_CSR is read-write
1: OPAMP_CSR is read-only

Bit 30 **CALOUT:** Operational amplifier calibration output

This bit reflects the digital value of OPAMP output when OPAINTOEN bit is low. During the input offset calibration procedure, it provides the indication that the target trimming value is reached, by transiting from 1 to 0.
The bit cannot be used as long as the OPAINTOEN bit is high.

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **TRIMOFFSETN[4:0]**: Trim for NMOS differential pairs

Upon reset, this bitfield is loaded with the factory trimming value. It can only be modified when the USERTRIM bit is set.
0x00 Reserved (should not be used for the triming value)
0x01 minimum triming value
0x1F maximum triming value

Bits 23:19 **TRIMOFFSETP[4:0]**: Trim for PMOS differential pairs

Upon reset, this bitfield is loaded with the factory trimming value. It can only be modified when the USERTRIM bit is set.
0x00 Reserved (should not be used for the triming value)
0x01 minimum triming value
0x1F maximum triming value

Bits 18:14 **PGA_GAIN[4:0]:** Operational amplifier programmable gain and PGA flavor primary control
This bitfield selects the programmable gain (bits [2:1]) and the PGA flavor ([bits 4:3]). Positive numbers represent positive gain (noninverting PGA), negative numbers negative gain (inverting PGA).
00000: gain 2
00001: gain 4
00010: gain 8
00011: gain 16
01000: gain -1 / gain 2 with bias on OPAMP_VINM0
01001: gain -3 / gain 4 with bias on OPAMP_VINM0
01010: gain -7 / gain 8 with bias on OPAMP_VINM0
01011: gain -15 / gain 16 with bias on OPAMP_VINM0
10000: gain 2 with filtering on OPAMP_VINM0
10001: gain 4 with filtering on OPAMP_VINM0
10010: gain 8 with filtering on OPAMP_VINM0
10011: gain 16 with filtering on OPAMP_VINM0
11000: gain -1 / gain 2 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11001: gain -3 / gain 4 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11010: gain -7 / gain 8 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11011: gain -15 / gain 16 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
Other: Not used

Bits 13:12 **CALSEL[1:0]:** Calibration selection
This bitfield selects the offset calibration reference voltage applied on the operational amplifier inputs when CALON and FORCE_VP bits are set.
00: $0.033*V_{DDA}$
01: $0.1*V_{DDA}$ (for PMOS calibration)
10: $0.5*V_{DDA}$
11: $0.9*V_{DDA}$ (for NMOS calibration)

Bit 11 **CALON:** Calibration mode enable
This bitfield disables or enables the calibration mode.
0: Calibration mode disabled
1: Calibration mode enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **OPAINTOEN:** Operational amplifier internal output enable
This bit determines where the operational amplifier output is connected, either to the OPAMP_VOUT pin or internally to an ADC/COMP channel.
0: OPAMP_VOUT pin
1: ADC/COMP channel

Bit 7 **OPAHSM:** Operational amplifier high-speed mode
This bit determines the speed (slew rate) of the operational amplifier.
0: Normal speed
1: High speed
Only change this configuration when OPAMP is disabled.

Bits 6:5 **VM_SEL[1:0]:** Inverting input primary selection
This bitfield determines where the operational amplifier inverting input is connected to.
00: OPAMP_VINM0 pin
01: OPAMP_VINM1 pin
10: Internal resistor divider middle point
11: Operational amplifier output
The setting 10 is for the PGA use cases. With this setting, the connection of the OPAMP_VINM0 and OPAMP_VINM1 pins is determined by the PGA_GAIN[4:0] bitfield.
The setting 11 sets OPAMP in follower mode. With the other two settings, OPAMP is configured as a stand-alone operational amplifier.

Bit 4 **USERTRIM**: User trimming enable
This bit enables the user trimming of the operational amplifier input offset. When set, the TRIMOFFSETN[4:0] and TRIMOFFSETP[4:0] bitfields, loaded with factory trimming values upon reset, become writable. This allows modifying the input offset trimming values by the user.
0: Disable (trimming by user not possible)
1: Enable (trimming by user possible)

Bits 3:2 **VP_SEL[1:0]:** Noninverting input primary selection
This bitfield determines where the operational amplifier noninverting input is connected to.
00: OPAMP_VINP0 pin
01: OPAMP_VINP1 pin
10: OPAMP_VINP2 pin
11: OPAMP_VINP3 pin or DAC output channel
Refer to *Section 31.3: OPAMP implementation* for OPAMP instance-dependent information.

Bit 1 **FORCE_VP:** Force internal reference on noninverting input
This bit is reserved for test. It allows forcing the calibration reference voltage selected through the CALSEL[1:0] bitfield on the operational amplifier noninverting input.
0: Do not force (normal operation)
1: Force

Bit 0 **OPAEN:** Operational amplifier enable
This bit enables and disables the operational amplifier.
0: Disable
1: Enable

31.6.2 OPAMP timer-controlled mode register (OPAMP_TCMR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMPGA_SEL[2:0]			PGAS_GAIN[4:0]					Res.	Res.	TIMCM_SEL[2:0]			VPS_SEL[1:0]		VMS_SEL
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: OPAMP_TCMR register lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.
This bit determines whether the OPAMP_TCMR register is read-write or read-only.
0: Read-write
1: Read-only (OPAMP_TCMR locked)

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:13 **TIMPGA_SEL[2:0]**: Timer toggle signal selection for programmable gain control

This bitfield is set and cleared by software. It selects the signal (coming from timer to OPAMP multiplexers) that toggles the programmable gain control bitfields between the primary (PGA_GAIN[4:0], signal at low level) and secondary (PGAS_GAIN[4:0], signal at high level).
000: None (gain permanently controlled through PGA_GAIN[4:0] and VM_SEL[1:0])
001: opamp_tp1
010: opamp_tp2
011: opamp_tp3
100: opamp_tp4
101: opamp_tp5
110: opamp_tp6
111: opamp_tp7

Bits 12:8 **PGAS_GAIN[4:0]**: Operational amplifier programmable gain and PGA flavor secondary control
This bitfield selects the programmable gain (bits [2:1]) and the PGA flavor ([bits 4:3]) when the timer signal selected through TIMPGA_SEL[2:0] is high. Positive numbers represent positive gain (noninverting PGA), negative numbers negative gain (inverting PGA).
00000: gain 2
00001: gain 4
00010: gain 8
00011: gain 16
01000: gain -1 / gain 2 with bias on OPAMP_VINM0
01001: gain -3 / gain 4 with bias on OPAMP_VINM0
01010: gain -7 / gain 8 with bias on OPAMP_VINM0
01011: gain -15 / gain 16 with bias on OPAMP_VINM0
10000: gain 2 with filtering on OPAMP_VINM0
10001: gain 4 with filtering on OPAMP_VINM0
10010: gain 8 with filtering on OPAMP_VINM0
10011: gain 16 with filtering on OPAMP_VINM0
11000: gain -1 / gain 2 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11001: gain -3 / gain 4 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11010: gain -7 / gain 8 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
11011: gain -15 / gain of 16 with bias on OPAMP_VINM0 and filtering on OPAMP_VINM1
Other: Not used

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:3 **TIMCM_SEL[2:0]**: Timer toggle signal selection for operational amplifier input control
This bitfield is set and cleared by software. It selects the signal (coming from timer to OPAMP multiplexers) that toggles the operational amplifier input configuration control bitfields between the primary (VP_SEL[1:0] / VM_SEL, signal at low level) and secondary (VPS_SEL[1:0] / VMS_SEL, signal at high level).
000: None (input configuration permanently controlled through VP_SEL[1:0] and VM_SEL)
001: opamp_tc1
010: opamp_tc2
011: opamp_tc3
100: opamp_tc4
101: opamp_tc5
110: opamp_tc6
111: opamp_tc7

Bits 2:1 **VPS_SEL[1:0]**: OPAMP noninverting input secondary selection.
This bitfield is set and cleared by software. It selects the signal to connect to the operational amplifier noninverting input as long as the level of the timer signal selected through the TIMCM_SEL[2:0] bitfield is high.
00: OPAMP_VINP0
01: OPAMP_VINP1
10: OPAMP_VINP2
11: OPAMP_VINP3 pin or DAC output channel
Refer to *Section 31.3: OPAMP implementation* for OPAMP instance-dependent information.

Bit 0 **VMS_SEL**: OPAMP inverting input secondary selection

This bitfield is set and cleared by software. It selects the signal to connect to the operational amplifier inverting input as long as the level of the timer signal selected through the TIMCM_SEL[2:0] bitfield is high. The effect also depends on the VM_SEL[1:0] bitfield.

Condition: VM_SEL[1:0] = 00 or 01

0: OPAMP_VINM0 (operation as stand-alone operational amplifier)

1: OPAMP_VINM1 (operation as stand-alone operational amplifier)

Condition: VM_SEL[1:0] = 10 or 11

0: Internal resistor divider middle point (operation as PGA)

1: Operational amplifier output (operation as follower)

31.6.3 OPAMP register map

Table 274. OPAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**OPAMP_CSR**	LOCK	CALOUT	Res.	TRIMOFFSETN [4:0]					TRIMOFFSETP [4:0]					PGA_GAIN[4:0]					CALSEL	CALON	Res.	Res.	OPAINTOEN	OPAHSM	VM_SEL		USERTRIM	VP_SEL[1:0]		FORCE_VP	OPAEN
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0
0x004	**OPAMP_TCMR**	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMPGA_SEL [2:0]		PGAS_GAIN [4:0]						Res.	Res.	TIMCM_SEL [2:0]			VPS_SEL[1:0]		VMS_SEL
	Reset value	0																0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x008 - 0x3FC	**Reserved**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.3 on page 122* for the register boundary addresses.

32 Coupling and chaining bridge (CCB)

In this section, any references to CCB only apply to STM32G56x products. The STM32G53x/54x/55x products do not have the CCB peripheral.

32.1 CCB introduction

The coupling and chaining bridge (CCB) is a sophisticated security mechanism designed to ensure that private keys are protected, even against CPU access, and are wrapped with unique device keys, and stored securely in a secure storage.

The CCB can be programmed to implement special coupling and chaining operations required to protect private keys used in PKA protected operations.

These coupling and chaining operations involve the PKA, the SAES, and sometimes the RNG peripherals.

32.2 CCB main features

The principal features of the CCB are listed below:

- AHB system slave port, mapping multiple peripherals (see *Section 32.3*)
- AHB configuration slave port (CCB peripheral), accessible in write through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored). There is no limitation for reads.
- Optional coupling of PKA RAM writes to SAES input data register
- Support for multiple optional chaining options:
  - Read and write chaining between PKA RAM and SAES
  - SAES to CCB read chaining, with comparison to a 128-bit reference tag
  - RNG output chaining with either PKA RAM or SAES_IVR registers
- Dedicated sequences to support three PKA protected operations (modular exponentiation, scalar multiplication, ECDSA signature)
  - One-time sequence to prepare a PKA protected operation (blob creation)
  - Many-time sequence to execute PKA protected operation (blob usage)
- Optional private key generation for ECDSA signature, and for ECC private key cryptography (key never accessible to the application in clear-text)
- Possibility to encrypt, with AES-256, any PKA blob encryption key using the device unique secret key DHUK (in the SAES), which makes the PKA encrypted blob usable only on this device
- Software-reset capability
- Repository for cryptographic subsystem tamper event flags

32.3 CCB implementation

The CCB connects the peripherals listed in *Table 275*.

Table 275. Peripherals connected by CCB interconnect

Peripheral	Coupling and chaining feature(1)	Tamper status flag in CCB_SR
RNG	Yes	TAMP_EVT0
SAES	Yes	TAMP_EVT1
PKA	Yes	TAMP_EVT2 (registers + PKA RAM)
		TAMP_EVT3 (PKA operations)
CCB	Yes	TAMP_EVT4
AES	No	TAMP_EVT5
HASH	No	-

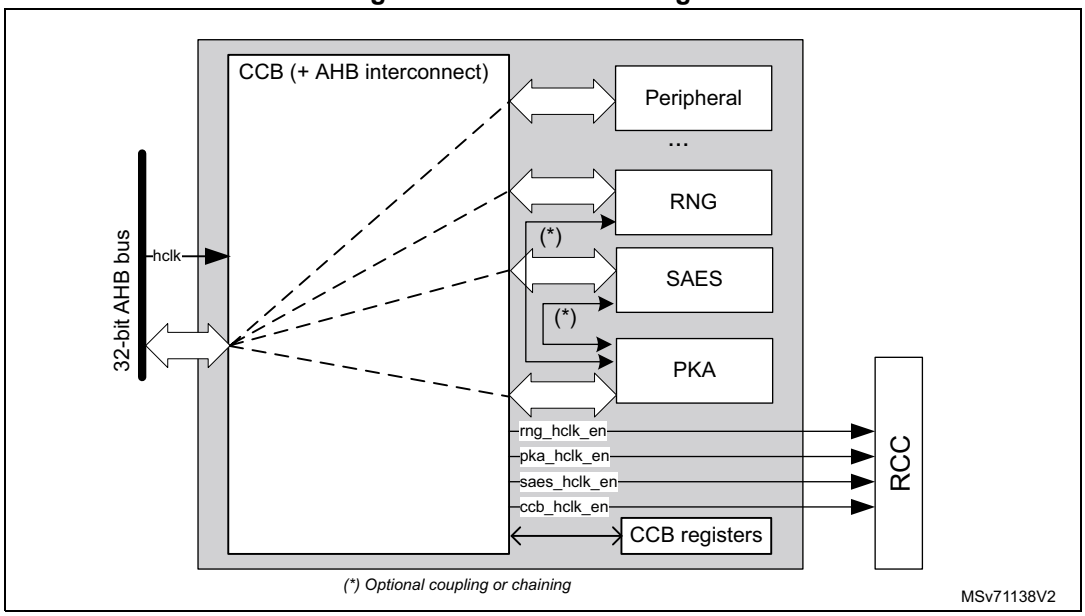
1. See *Section 32.4.4* for details.

32.4 CCB functional description

32.4.1 CCB block diagram

Figure 295 shows the CCB block diagram. Peripherals connected to the CCB interconnect, with optional coupling/chaining features, are defined in *Section 32.3*.

Figure 295. CCB block diagram



32.4.2 CCB internal signals

Table 276 describes the relevant internal signals interfacing the CCB peripheral. Signals targeting peripherals connected to the CCB are not listed.

Table 276. CCB internal input/output signals

Signal name	Signal type	Description
hclk	Digital input	Interconnect system clock input
ccb_hclk_en	Digital output	CCB clock request to the RCC
x_hclk_en	Digital output	Peripheral x clock request to the RCC (x = SAES, PKA, or RNG)

32.4.3 CCB reset and clocks

The CCB shares the same clock used by the AHB bus. Once the CCB clock is activated in the RCC, and coupling and chaining activity is activated by setting CCOP ≠ 0 in CCB_CR, the CCB automatically requests the RCC to enable RNG, SAES, and PKA clocks . These clocks remain enabled until the application gates them in the RCC, with CCOP cleared in the CCB_CR register.

The CCB has a dedicated reset bit in the RCC, and a system reset input. Locally to the CCB it is also possible to trigger a reset of the peripheral based on IPRST (do not use it when the application does not use CCB)in the CCB_CR register (refer to the bitfield description for details). Status register CCB_SR has a specific reset behavior (see *Section 32.7.2*).

When at least a cryptographic peripheral raises a tamper event, the CCB requests its clock to the RCC. The tamper status TAMP_EVTi of these peripherals can be found in the CCB_SR register, as defined in *Section 32.3*.

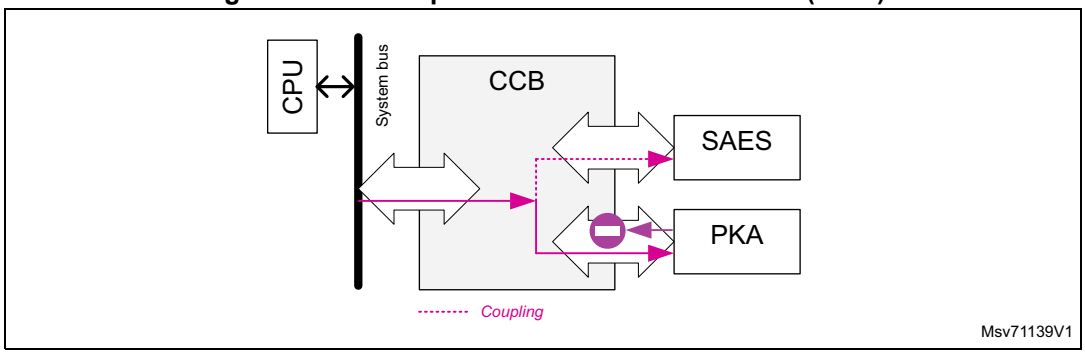
32.4.4 CCB coupling and chaining

Note: The selection of these chaining modes is defined in Section 32.5. In most of these operations, a read of the PKA RAM returns 0 (shown with a purple circle in the following figures).

Coupling CPU → PKA + SAES (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and sends them to SAES_DINR, as shown in *Figure 296*. Write to SAES_DINR is not coupled to the PKA. The CCB sends both data and address to SAES_DINR.

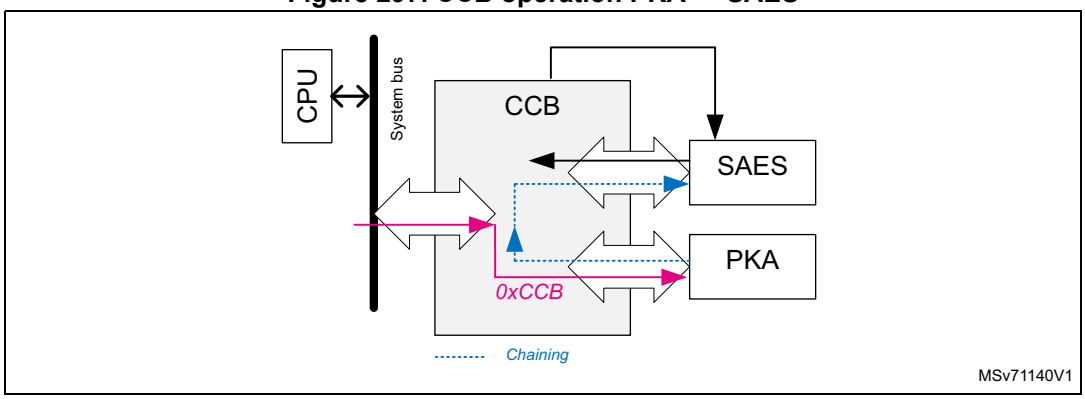
Figure 296. CCB operation CPU → PKA + SAES (write)



Chaining PKA → SAES

When the corresponding OPSTEP is selected, for each 32-bit write request to the PKA RAM of magic number 0xCCB, the CCB generates a read request from the write target (in the PKA RAM) to SAES_DINR.

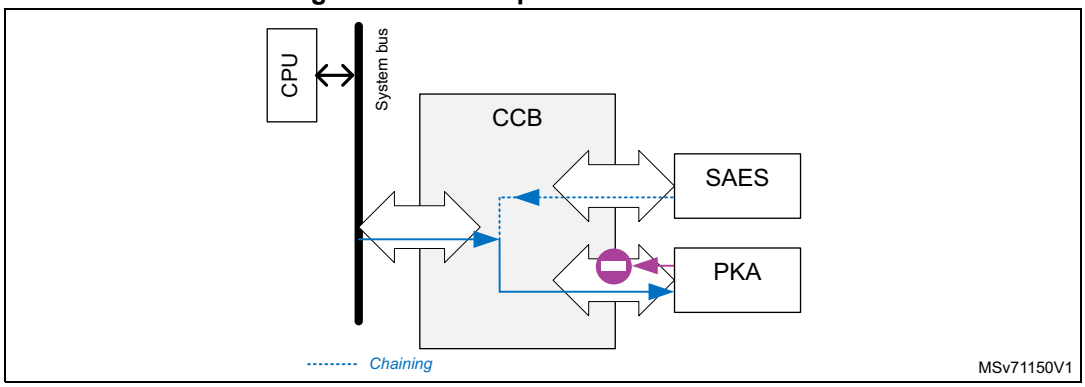
Figure 297. CCB operation PKA → SAES



Chaining SAES → PKA

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and substitutes these data by a 32-bit word coming from SAES_DOUTR. When available, this chaining is activated when the application writes 0xCCB to PKA RAM.

Figure 298. CCB operation SAES → PKA

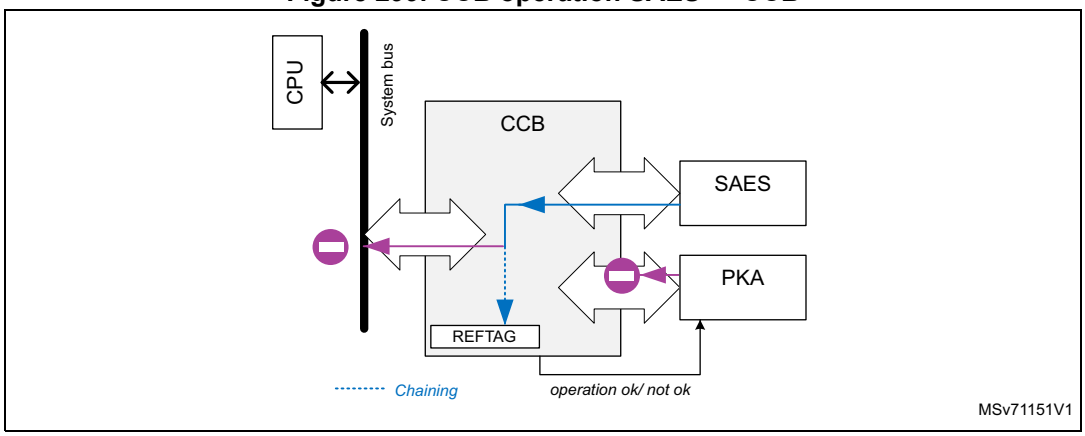


Chaining SAES → CCB

When the corresponding OPSTEP is selected, the CCB takes the read requests to SAES_DOUTR, and sends them to the system interconnect and to CCB_REFTAGRx holding the reference TAG information. For confidentiality reason, read data toward system bus are zeroed by the CCB.

If the CCB finds that the transfered value matches its reference information, it signals to the PKA that it can proceed with the protected PKA operation (START bit can be set).

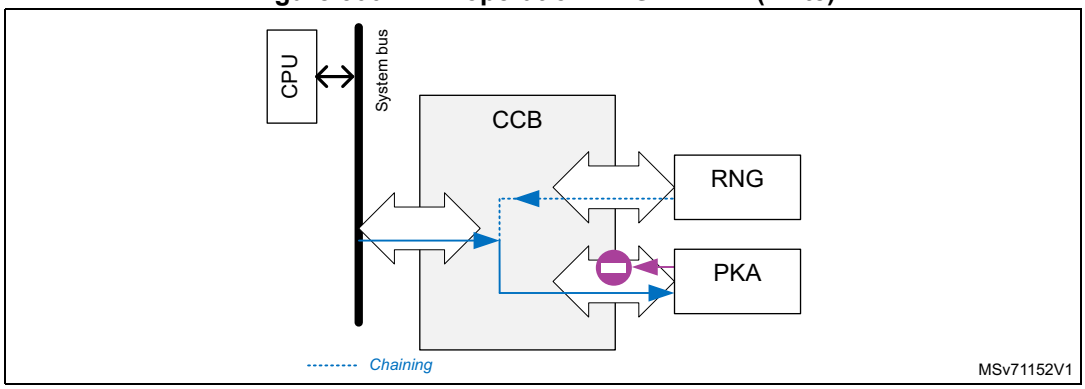
Figure 299. CCB operation SAES → CCB



Chaining RNG → PKA (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and substitutes these data with random numbers coming from RNG_DR. For consistency, the application can write 0xCCB to activate this chaining.

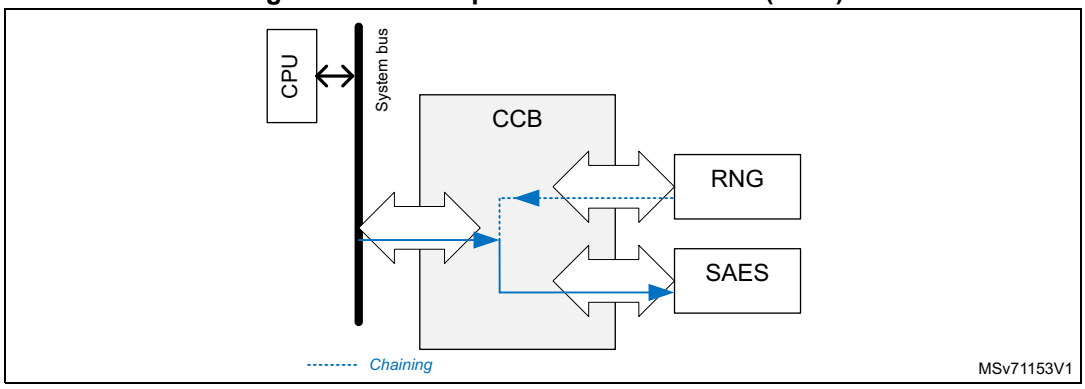
Figure 300. CCB operation RNG → PKA (write)



Chaining RNG → SAES (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to SAES_IVR1 to IVR3, and substitutes these data with random numbers coming from RNG_DR. For consistency, the application can write 0xCCB to activate this chaining.

Figure 301. CCB operation RNG → SAES (write)



32.4.5 CCB error management

When the CCB detects an error during a coupling and chaining operation (see *Section 32.5*), it sets accordingly OPERR[5:0] in CCB_SR (see *Section 32.7.2* for the possible errors).

When OPERR[5:0] ≠ 0 in CCB_SR, the CCB raises IPRST in CCB_CR, sets OPSTEP[4:0] to 0x1F, and sets CCB_BUSY bit to 1. The application must clear IPRST to be able to use again the CCB and the peripheral(s) used by the coupling and chaining operation.

32.5 CCB coupling and chaining operations

When an application must enforce private key confidentiality by wrapping them with a 256-bit hardware unique key, the following steps are needed:

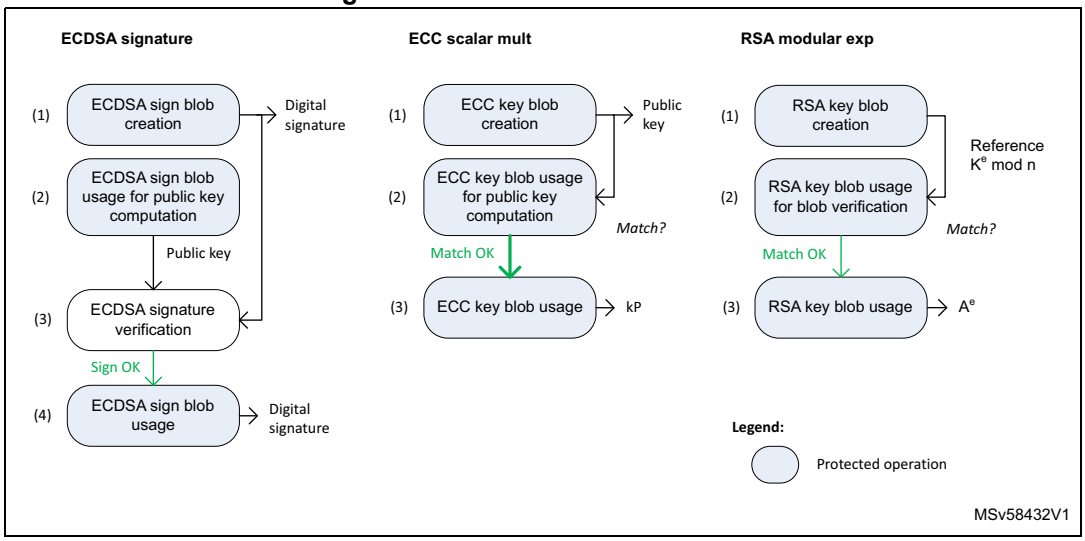
1. Select the protected operation in the PKA that uses this private key (ECDSA signature, scalar multiplication, modular exponentiation). Then, choose the CCOP value in the CCB that corresponds to the selected operation, as defined in this section. Then, as multiple cryptographic peripherals are used, make sure to follow the guidelines described in *Section 32.5.1*
2. When CCB is active (OPSTEP=0x1), select in the SAES the DHUK (with optional XOR with BHK), or unwrap in the SAES an existing blob encryption key that has been encrypted (in wrapped-key mo.de) using the DHUK. A 256-bit KEYSIZE must be selected.
   As the DHUK value is different when CCB is active or inactive, the operation of wrapping or unwrapping blob encryption key in SAES (with KMOD[1:0]=0x1) must be done when OPSTEP=0x1. Refer to operation with wrapped keys in *Section 35: Secure AES coprocessor (SAES)* for details.
3. When the application uses this protected operation in the PKA, it must load the related blob information in the PKA RAM and in SAES_DINR.
   – If the information exactly matches the conditions of the blob generation (choice and order of the writes of protected parameters in the PKA RAM, selection of blob

encryption key, use of original encrypted secrets), the CCB allows the application to set START bit in PKA_CR.

– If the blob information does not match the conditions of the blob generation, the application can only reset the sequence by setting IPRST bit in CCB_CR. CCB, PKA, and SAES become available when CCB_BUSY is cleared in CCB_SR.

4. At any time, the application can abort the executing protected operation by using the IPRST bit in CCB_CR.
5. When creating a blob, always use it once before trusting it to be stored in non-volatile memory or given to a less trusted code (see *Figure 302*). More specifically, each time a blob is created:
   – For ECDSA signature blob: use it to compute the public key, and use this key to verify the signature computed during the blob creation.
   – For ECC key blob: use it to compute the public key, and verify key coordinates match those of the key computed during the blob creation.
   – For RSA key blob: use it to compute a reference modular exponentiation and verify it matches the one computed during the blob creation.

Figure 302. Blob creation scenarios



This section describes how to do the above steps, for all available protected operations in the PKA. Note that:

- **Bold** elements in the following tables are key inputs that the application must save in non-volatile memory after the newly created blob has been confirmed by its first usage.
- The CCB automatically goes to the next step, updating the OPSTEP value, when the application clears or performs the end-step event described in tables of *Section 32.5.2* to *Section 32.5.4*.
- When an incremental read or write is mentioned, the application must write four bytes in the PKA memory, with an incremental memory address (modulo 0x8). Otherwise, the CMF error flag is triggered, and the process must be restarted.
- When writing to SAES_CR register the application must write the configuration expected by the CCB, otherwise an OPERR might be raised. Refer to the OPSTEP “CPU action” instructions in the application tables (*Table 277* through *Table 283*).
- When the CCB uses the RNG, the application must check in RNG_SR that no error occurred (SEIS = CEIS = 0), and that a random number is ready (DRDY = 1). In case of issue, RNGOKF is not set, and the application must ensure that the RNG is functional before running again the OPSTEP sequence that uses the RNG.
- For each OPSTEP>0, the application must manage the error flags that could occur in the RNG, SAES or PKA blocks. During a normal protected operation, no error flags are expected. Refer to *Section 32.4.5: CCB error management* for details on CCB error management.
- When CCB draws random number to compute ECC private keys, it implements the testing candidates method defined by NIST in FIPS PUB 186-4 section B.4.

Note: It is not possible to generate RSA key pairs with the CCB, they must be generated outside the device.

32.5.1 Preliminary steps before any protected operation

The following steps are required before running any operation described in this section:

1. Initialize the GTZC using the same security/privilege configuration protections for the SAES, PKA, RNG and CCB peripherals (see *Section 5: Global TrustZone® controller (GTZC)*). Otherwise OPERR = 0x1A occurs (see *Section 32.7.2*).
2. Activate both RNG clocks, then initialize the peripheral to be fully functional, without pending errors.
3. Activate the PKA clock, then release the PKA reset. The PKA is functional once its RAM is automatically erased, and it has fetched the necessary random numbers. For further details refer to enabling/disabling PKA in *Section 37: Public key accelerator (PKA)*.
4. Activate the SAES clock, then release the SAES reset. The SAES is functional once it has fetched necessary random numbers.
5. Activate the CCB clock, set IPRST bit in CCB_CR, then clear it when CCB_BUSY = 0x0 in CCB_SR.

Note: Step 4 can be done before step 3, as CCOP = 0x0.

32.5.2 ECDSA signature

This PKA operation is protected against side-channel attacks. When CCOP = 0x0, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0xC3, the private key d can be manipulated by the application only if encrypted, using a special blob called “ECDSA signature blob”.

ECDSA signature blob creation

The ECDSA signing operation is prepared by generating and then encrypting the private key d, and computing the reference GCM tag. This process is summarized *Table 277*, with coupling and chaining operations described in *Section 32.4.4*. This sequence starts by setting CCOP to 0xC0 or 0xC2 in CCB_CR.

Table 277. Blob creation steps for ECDSA signature (CCOP = 0b110000<i>0)

OP STEP	MODE in PKA	GCMPH in SAES	Coupling or chaining	End-step event(1)	CPU action in the step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x24 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt the blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set AES-GCM up in 256-bit encryption mode, with the DHUK or the decrypted key in register (if applicable). – Write hash z in the PKA RAM.
0x2		0x0	RNG -> SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. SAES_IVR1 to SAES_IVR3 are loaded with randoms. – Read back **blob IV** in SAES_IVR registers.
0x3	0x24	0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in the SAES triggered after each 2x32-bit write.
0x6 (<i>=1)	0x24	0x2	RNG -> PKA	RNGOKF set	Write incrementally d in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if sanity check fails).
0x4 (<i>=0)			n/a	DATAOKF set	Write incrementally d in the PKA.
0x8	0x24		PKA -> SAES	DATAOKF set	– Read incrementally d from the PKA (writing 0xCCB magic numbers), and run AES encryption rounds. Special rule must be used for the last 64-bit word (see step 7 after the table). – Read **encrypted d** in SAES_DOUTR. – Repeat for each 16 bytes block of data.
0x9			RNG -> PKA	RNGOKF set, GCMPH = 0x3	– Write incrementally k in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if the sanity check fails).
0xA(2)		0x3	N/A	START= 0x1	– Complete GCM final. – Read **reference tag** in the SAES. – Start the operation in PKA setting START bit in PKA_CR For the first two steps keep the SAES enabled (EN bit set).

Table 277. Blob creation steps for ECDSA signature (CCOP = 0b110000<i>0) (continued)

OP STEP	MODE in PKA	GCMPH in SAES	Coupling or chaining	End-step event(1)	CPU action in the step
0x19(3)	0x24	N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM reads as 0
0x1A		N/A	N/A	IPRST set in the CCB	– Read signature (r,s) in PKA RAM, and save it in SRAM. It is used (at least once) for blob creation verification. – Set IPRST in CCB
0x0	N/A	N/A	N/A	N/A	– Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR. – Verify the newly created blob, running the *ECDSA signature blob use for public-key computation* with it.

1. The application must clear and/or perform the described event for the CCB to automatically go to the next step.
2. Once START bit is set SAES is reset, and OPERR linked to SAES are deactivated. Also when START bit is set, CCB sets OPSTEP to 0x19 after a few cycles.
3. SAES cannot be used for any operations, for security reasons.

The sequence described in *Table 277* is detailed below. Refer to the table for OPSTEP values and related coupling/chaining modes.

1. Write CCOP = 0xC0 or 0xC2 in CCB_CR.
2. Enable the PKA in the correct mode, writing in PKA_CR MODE=0x24 and EN=1 (repeat if the EN bit stays clear). Wait for INITOK=1 in PKA_SR.
3. Write the hash of the message in the PKA RAM
4. Initialize the ECC signature blob key in the SAES. This key can be the DHUK (KEYSEL = 0x1 or 0x4), or a key decrypted in 256-bit wrapped key mode by the DHUK (KEYSEL = 0x1 or 0x4, KMOD = 0x1, MODE = 0x1 or 0x2). See *Section 35: Secure AES coprocessor (SAES)* for details.
5. In the SAES, set the GCM chaining mode in encryption mode (CHMOD = 0x3, GCMPH = 0x0, MODE = 0x0, DATATYPE = 0x0). Then:
   a) Configure the 256-bit key size using KEYSIZE.
   b) Write SAES_IVRx, with any value (except SAES_IVR0 that must be equal to 0x2). For SAES_IV1 to IV3, random values are loaded into the SAES by the CCB. If an error occurs, RNGERRF is set in CCB_SR.
   c) Read back SAES_IVR1 to SAES_IVR3, and **save the value** in non-volatile memory after blob verification.
   d) Set EN in SAES_CR, and wait until CCF = 1 in SAES_ISR (end of GCM init).
   e) Clear CCF flag in SAES_ICR.
6. Select the header phase of the running GCM chaining mode (GCMPH = 0x1). Write the protected operation (ECDSA signature) mandatory parameters in the PKA RAM, as described in *Section 37: Public key accelerator (PKA)*. Do the following when writing data in the PKA:
   a) Write two 32-bit word in the PKA RAM, then wait until CCF is set in SAES_ISR. An SAES interrupt can be triggered when this flag is set.
   b) Clear CCF in SAES_ICR.

7. Select the payload phase of the running GCM chaining mode (GCMPH = 0x2). Write the private key *d* in the PKA RAM using the following sequence:
   a) If CCOP = 0xC0, write the private key d value in the PKA RAM. If CCOP = 0xC2, write the 32-bit magic number 0xCCB to the private key d parameter addresses in the PKA memory. The CCB then draws the value of *d* from the RNG, with PKA running a special sanity check to verify random compliance with NIST standard.
   b) Complete parameter information with the mandatory two 32-bit zeroed words. At the end of this process, the private key d of size *d_size* is written in the PKA RAM. Clear the error flag, then restart step 6 if the PKA sets RNGERRF when BUSY is cleared.
8. When the PKA sets its completion flag (RNGOKF if CCOP = 0xC2, DATAOKF if CCOP = 0xC0), clear it, then read the private key d in the PKA RAM using the sequence below (starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8):
   a) For each 64-bit word in the PKA RAM, write twice the 32-bit magic number 0xCCB to the private key d parameter addresses in the PKA memory. The target content of the PKA memory is transfered to SAES_DINR by the CCB. Wait until CCF is set in SAES_ISR (end of AES encryption). An SAES interrupt can be triggered when this flag is set.
   b) Clear CCF flag in SAES_ICR. Then read the encrypted private key d in SAES_DOUT, **to save it** in the non-volatile memory after blob verification.
   c) For the final 64-bit word (mandatory zeroed word): if the *d_size* is proportional to 16 bytes (for example, for 192- and 521-bit curves), the final 64 bits must be written to with two 32-bit magic numbers 0xCCB. If the *d_size* is not proportional to 16 bytes (for example, for 256- and 512-bit curves), the final 64 bits must be written with 0s.
9. When DATAOKF = 1 in PKA_SR (private key d ok), clear this flag. Then write to the integer k parameter addresses in the PKA memory, starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8. The CCB draws the value of k from the RNG. The PKA then runs a special sanity check to verify random compliance with NIST standard. Clear the error flag then restart this step if the PKA sets RNGERRF when BUSY is cleared.
10. When RNGOKF = 1 in PKA_SR (random k OK), clear this flag. Then select the final phase of the running GCM chaining mode (GCMPH = 0x3). Write the last GCM block in SAES_DINR. Then wait until CCF is set in SAES_ISR (end of GCM final).
11. Read SAES_DOUTR four times to retrieve the ECDSA signature blob reference tag. Save it in the non-volatile memory after blob verification. Do not disable SAES by clearing the EN bit.
12. Set START bit in PKA_CR to run the ECDSA signature operation.
13. When the PROCENDF completion flag is set save the ECDSA signature (r,s) of the hash z in SRAM to use it after using the new blob for public-key computation.
14. Clear CCOP by setting IPRST bit in CCB_CR. After CCB_BUSY is cleared in CCB_SR, clear the IPRST bit. The SAES, and PKA can again be used for any tasks. They are unprotected by the CCB until CCOP is written different from 0.

ECDSA signature blob usage for ECDSA signature

Table 278 summarizes the ECDSA signing operation using the encrypted private key d, the reference blob IV, and the computed GCM tag of *Table 277*. Coupling and chaining traffic options are described in *Section 32.4.4*. This sequence starts by setting CCOP to 0xC3 in the CCB_CR register.

Table 278. Blob use steps for ECDSA signature (CCOP = 0b11000011)

OPSTEP	MODE in PKA	GCMPH	Coupling or chaining operations	End-step event(1)	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x24 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt the blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set AES-GCM up in 256-bit encryption mode, with the DHUK or the decrypted key in register (if applicable). – Write hash z.
0x12		0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13	0x24	0x1	CPU –> PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in the SAES, triggered after each 64-bit write.
0x14		0x2	SAES –> PKA	DATAOKF set	– Write encrypted d in SAES_DINR, and runs AES decryption round. – Write incrementally private key d in the PKA, using 0xCCB magic number. A special rule must be used for the last 64-bit write (see step 7 in the sequence detailed after this table). – Repeat for each 16-byte block of data.
0x16			RNG –> PKA	RNGOKF set, GCMPH = 0x3	Write incrementally k in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if the sanity check fails).
0x17		0x3	SAES –> CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x24	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)		N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM is read as 0.
0x1A		N/A	N/A	N/A	Read **signature (r,s)** in the PKA RAM.

1. The application must clear and/or perform the described event for the CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons..

The sequence described in *Table 278* is detailed below. Refer to the table for the OPSTEP values and related coupling and chaining modes.

1. Write CCOP = 0xC3 in CCB_CR.
2. Enable the PKA in the correct mode, writing in PKA_CR MODE=0x24 and EN=1(repeat if the EN bit stays clear). Wait for INITOK=1 in PKA_SR.
3. Write the hash of the message in the PKA RAM.
4. Initialize the ECC signature blob key in the SAES. This key can be the DHUK (KEYSEL = 0x1 or 0x4), or a key decrypted in 256-bit wrapped key mode by the DHUK (KEYSEL = 0x1 or 0x4, KMOD = 0x1, MODE = 0x1). Refer to *Section 35: Secure AES coprocessor (SAES)* for details.
5. Set the GCM chaining mode in decryption in the SAES (CHMOD = 0x3, GCMPH = 0x0, MODE = 0x2, DATATYPE = 0x0). Then:
   a) Configure the 256-bit key size using KEYSIZE.
   b) Write to SAES_IVRx registers the value saved in non-volatile memory.
   c) Set EN in SAES_CR, and wait until CCF is set in SAES_ISR (end of GCM init).
   d) Clear CCF flag in SAES_ICR.
   e) Write to CCB_REFTAGR registers, the ECDSA signature blob reference tag value saved in non-volatile memory.
6. Select the header phase of the running GCM chaining mode (GCMPH = 0x1). Write the protected operation (ECDSA signature) mandatory parameters in the PKA RAM, exactly as done during the blob creation. See step 5 in *ECDSA signature blob creation* for details.
7. Select the payload phase of the running GCM chaining mode (GCMPH = 0x2). Write the protected private key d in the PKA RAM, following the below sequence (starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8):
   a) Writetwo 64-bit words of encrypted private key d in SAES_DINR. Wait until CCF is set in SAES_ISR (end of AES decryption). An SAES interrupt can be triggered when this flag is set.
   b) Clear the CCF flag in the SAES_ICR register. Then, for both 64-bit words, write four times the 32-bit magic number 0xCCB to the private key d parameter addresses in the PKA memory. The CCB transfers the clear-text content of SAES_DOUTR to the PKA.
   c) For the final 64-bit word (mandatory zeroed word): If *d_size* defined in step 6 in *ECDSA signature blob creation* is not proportional to 16-bytes (for example, for 256- and 512-bit curves), the final 64 bits must be written with 0s.
8. When DATAOKF = 1 in PKA_SR (private key d ok), clear this flag. Then write to the integer k parameter addresses in the PKA memory, starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8. The CCB draws the value of k from the RNG. The PKA then runs a special sanity check to verify random compliance with NIST standard. Clear the error flag, then restart step 7 if the PKA sets RNGERRF when BUSY is cleared.
9. When RNGOKF = 1 in PKA_SR (random k ok), clear this flag. Then select the final phase of the running GCM chaining mode (GCMPH = 0x3). Write the last GCM block in SAES_DINR. Then wait until CCF is set in SAES_ISR (end of GCM final).
10. Read four times SAES_DOUTR (returns 0s). If PKA parameters were written exactly like during the blob creation, and the correct private key d was used, and all expected DATAOKF/RNGOKF flags were raised then cleared, the application can set START bit

in PKA_CR to run the ECDSA signature operation. Otherwise, IPRST in CCB_CR must be used to restart the sequence from step 1.

11. Clear CCOP by setting IPRST in CCB_CR. After CCB_BUSY is cleared in CCB_SR, clear the IPRST bit. The SAES, and PKA can again be used for any tasks. They are unprotected by the CCB until CCOP in CCB_CR is written different from 0.

ECDSA signature blob use for public-key computation

The ECC public key computation using the encrypted private key d, the reference blob IV, and the computed GCM tag of *Table 277* is summarized in *Table 279*. The additional ECDSA verification at the end of the sequence must be used before saving a new ECDSA signature blob in non-volatile memory. Shown coupling and chaining traffic options are described in *Section 32.4.4*. This sequence starts by setting CCOP to 0x81 in the CCB_CR register.

Table 279. Blob use steps for signing public key computation (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable).
0x12	0x20	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13		0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in SAES, triggered after each 2x32-bit write.
0x14		0x2	SAES -> PKA	DATAOKF set, GCMPH = 0x3	– Write encrypted d in SAES DINR and run AES decryption round. – Write incrementallyscalar multiplier k in the PKA, using 0xCCB magic number. – Repeat for each 16 bytes block of data.
0x17		0x3	SAES -> CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x20	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)		N/A	N/A	PROCENDF set	The PKA computes the operation. PKA RAM is read as 0.
0x1A	0x20	N/A	N/A	N/A	– Read **public key point Q** in PKA RAM

Table 279. Blob use steps for signing public key computation (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1A	0x20	N/A	N/A	IPRST set in the CCB	– Read public key point Q in PKA RAM – Set IPRST in CCB
0x0	0x26	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR. Before writing new ECDSA signature blob in non-volatile memory do the following: – Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x26 and EN=1. Wait for INITOK=1. – Write the hash z used during blob creation (see *Table 277*) – Write ECDSA verification parameters in PKA, including the public key coordinates computed in step 0x1A. – Start the operation setting START bit in PKA_CR – Wait for PROCENDF bit is set. If the signature is valid save the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons.

32.5.3 ECC scalar multiplication

This operation consists in the computation of k x P, where P is a point on a curve over prime fields, and “x” is the elliptic curve scalar point multiplication. The computation result is a point that belongs to the same curve, or a point at infinity.

The scalar multiplication operation is protected against side-channel attacks. When CCOP = 0x0 in CCB_CR, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0x81, the scalar multiplier k can be manipulated by the application only if encrypted, using a special blob called “ECC key blob”.

Scalar multiplication “ECC key” blob creation

The scalar multiplication operation is prepared by generating then encrypting the scalar multiplier k, and computing the reference GCM tag. This process is summarized in *Table 280*. Coupling and chaining operations described in *Section 32.4.4*. This sequence starts by setting CCOP to 0x80 or 0x82 in CCB_CR register.

Table 280. Blob creation steps for scalar multiplication (CCOP = 0b100000<i>0)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in 256-bit encryption mode with DHUK or the decrypted key in register (if applicable) – Write base point G coordinates as input in PKA RAM.
0x2		0x0	RNG –> SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. Registers IVR1 to 3 are loaded with randoms. – Read back **blob IV** in IVR registers.
0x3	0x20	0x1	CPU –> PKA + SAES	GCMPH = 0x2	Write ECC scalar multiplication parameters in the PKA RAM, clearing CCF flags in SAES, triggered after each 64-bit write.
0x6 (<i>=1)	0x20	0x2	RNG –> PKA	RNGOKF set	Write incrementally k in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if the sanity check fails).
0x4 (<i>=0)			N/A	DATAOKF set	Write incrementally k in the PKA.
0x8	0x20	0x2	PKA –> SAES	DATAOKF set, GCMPH = 0x3	– Read incrementally k from the PKA (writing 0xCCB magic numbers), and runs AES encryption rounds. – Read **encrypted k** in SAES DOUTR. – Repeat for each 16-byte block of data..
0xA(2)		0x3	N/A	START= 0x1	– Complete GCM final. – Read **reference tag** in the SAES. – Start the operation in PKA setting START bit in PKA_CR For the first two steps keep the SAES enabled (EN bit set).
0x19(3)	0x20	N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM reads as 0
0x1A		N/A	N/A	IPRST set in the CCB	– Read reference public key point Q in PKA RAM, and save it in SRAM to verify the new blob. – Set IPRST in CCB.
0x0	N/A	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. Once START bit is set SAES is reset, and OPERR linked to SAES are deactivated. Also when START bit is set, CCB sets OPSTEP to 0x19 after a few cycles.
3. SAES cannot be used for any operations, for security reasons.

Scalar multiplication blob use for example for ECC decryption

ECC scalar multiplication computation using the encrypted k, the reference blob IV and the computed GCM tag of *Table 280* is summarized in *Table 281*. Shown coupling and chaining traffic options are described in *Section 32.4.4*. This sequence starts by setting CCOP to 0x81 in CCB_CR register.

Table 281. Blob use steps for scalar multiplication (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally, decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable). – Write input point P coordinates in the PKA RAM. When computing a public key, write basepoint G coordinates.
0x12	0x20	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CBB_REFTAGR.
0x13		0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC scalar multiplication parameters in the PKA RAM, clearing CCF flags in the SAES, triggered after each 64-bit write.
0x14		0x2	SAES -> PKA	DATAOKF set, GCMPH = 0x3	– Write encrypted k in SAES_DINR, and runs AES decryption round. – Write incrementally scalar multiplier k in the PKA, using 0xCCB magic number. – Repeat for each 16 bytes block of data.
0x17		0x3	N/A	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x20	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR. Do not use SAES during this step (under reset).
0x19(2)				PROCENDF set	The PKA computes the operation. The PKA RAM is read as 0.
0x1A				N/A	Read **kP point coordinates** in the PKA RAM. If this point is the public key verify it matches the reference public key computed in *Table 280* before saving the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons.

32.5.4 RSA modular exponentiation

The modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of A^e mod n.

This modular exponentiation operation (MODE = 0x3) is protected against side-channel attacks. When CCOP = 0x0, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0x45, the exponent e and phi value can only be manipulated encrypted by application, using a special blob called “RSA key blob”.

Modular exponentiation “RSA key” blob creation

Modular exponentiation operation is prepared by generating then encrypting the scalar multiplier *k* and computing the reference GCM tag. This process is summarized in *Table 282*. The coupling and chaining operations described in *Section 32.4.4*. This sequence starts by setting CCOP to 0x44 in CCB_CR register.

Table 282. Blob creation steps for modular exponentiation (CCOP = 0b01000100)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x3 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in 256-bit encryption mode with DHUK or the decrypted key in register (if applicable). – Write a constant K as operand A in the PKA RAM.
0x2		0x0	RNG -> SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. SAES_IVR1 to 3 registers are loaded with randoms. – Read back blob IV in SAES_IVR registers.
0x3	0x03	0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write protected modular exponentiation parameters (PKA mode 0x3) in the PKA RAM, clearing CCF flags in the SAES triggered after each 2x32-bit write.
0x4	0x3	0x2	N/A	DATAOKF set	Write incrementally clear-text exponent e.
0x5				DATAOKF set	Write incrementally clear-text phi.
0x8			PKA -> SAES	DATAOKF set	– Read incrementallye from the PKA (writing 0xCCB magic numbers), and runs AES encryption rounds. – Read **encrypted e** in SAES_DOUTR. – Repeat for each 16 bytes block of data.
0x9				DATAOKF set, GCMPH = 0x3	– Read incrementally phi from the PKA (writing 0xCCB magic numbers), and run AES encryption rounds. – Read **encrypted phi** in SAES_DOUTR. – Repeat for each 16 bytes block of data
0xA(2)	0x3	0x3	N/A	START= 0x1	– Complete GCM final. – Read **reference tag** in the SAES. – Start the operation in PKA setting START bit in PKA_CR. For the first two steps keep the SAES enabled (EN bit set).

Table 282. Blob creation steps for modular exponentiation (CCOP = 0b01000100)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action in step
0x19(3)	0x3	N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM is read as 0
0x1A		N/A	N/A	IPRST set in the CCB	– Read the reference modular exponentiation K^e mod *n* in PKA RAM, and save it in SRAM to verify the new blob. – Set IPRST in CCB
0x0	N/A	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. Once START bit is set SAES is reset, and OPERR linked to SAES are deactivated. Also when START bit is set, CCB sets OPSTEP to 0x19 after a few cycles.
3. SAES cannot be used for any operations, for security reasons.

Modular exponentiation blob use (for example for RSA decryption)

The modular exponentiation computation using the encrypted *e* and *phi*, the reference blob IV, and the computed GCM tag of *Table 282* is summarized on *Table 283*. Shown coupling and chaining traffic options are described in *Section 32.4.4*. This sequence starts by setting CCOP to 0x45 in CCB_CR register.

Table 283. Blob use steps for modular exponentiation (CCOP = 0b01000101)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x3 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable). – Write operand A in the PKA RAM. Write constant K as operand A defined in *Table 282* when verifying the created blob.

Table 283. Blob use steps for modular exponentiation (CCOP = 0b01000101)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x12	0x03	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13		0x1	CPU –> PKA + SAES	GCMPH = 0x2	Write protected modular exponentiation parameters (PKA mode 0x3) in the PKA RAM, clearing CCF flags in the SAES, triggered after each -bit write.
0x14		0x2	SAES –> PKA	DATAOKF set	– Write encrypted e in SAES_DINR, and runs AES decryption round. – Write incrementally exponent e in the PKA, using 0xCCB magic number. – Repeat for each 16-byte block of data.
0x15				DATAOKF set, GCMPH = 0x3	– Write encrypted phi in SAES_DINR, and runs AES decryption round. – Write incrementally phi in the PKA, using 0xCCB magic number. – Repeat for each 16-byte block of data.
0x17		0x3	SAES –> CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x03	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)				PROCENDF set	The PKA computes the operation, PKA RAM is read as 0.
0x1A				N/A	Read $\mathbf{A^e}$ **mod** n in the PKA RAM. If the result is K^e mod n verify it matches the reference computed in *Table 282* before saving the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons.

32.6 CCB interrupts

The CCB does not support any interrupt.

32.7 CCB registers

The following section provides information on registers.

Reserved bits (marked as Res.) must be kept at their reset values, unless otherwise specified.

For general information relative to register description, refer to Document conventions.

These registers are accessible through the AHB interface of the CCB (32-bit words only). All reads are supported. Writes accesses must be 32-bit word only (other access types generate an AHB error).

32.7.1 CCB control register (CCB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Following a CCB hardware reset, the above reset value applies if OPERR[5:0] = 0x0 and no TAMP_EVT bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCOP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: CCB reset

Set this bit to start or abort any PKA protected operation defined in *Section 32.5*. IPRST bit must be low while writing to any CCB register.

After setting this bit, the application must clear it when CCB_BUSY and OPSTEP are cleared in CCB_SR (IPRST clear is ignored otherwise). After this clear all CCB registers are read with their reset valuesSetting IPRST clears any data stored in the SAES and the PKA.

When IPRST and CCB_BUSY are set, SAES registers are not accessible (read as 0, write ignored), and writes to EN and START bits in PKA_CR are ignored.

This bit is set by CCB when OPERR[5:0] ≠ 0 or any TAMP_EVT bit is set.

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **CCOP[7:0]**: Coupling and chaining operation

These bits must be set according to the chosen PKA protected operation (see *Section 32.5*).

0x044: Modular exponentiation blob creation with priv key from the CPU

0x045: Modular exponentiation blob use

0x080: ECC scalar multiplication blob creation with priv key from the CPU

0x082: ECC scalar multiplication blob creation with priv key from the RNG

0x081: ECC scalar multiplication blob use, or ECDSA signature blob use for public key computation

0x0C0: ECDSA signature blob creation with priv key from the CPU

0x0C2: ECDSA signature blob creation with priv key from the RNG

0x0C3: ECDSA signature blob use for ECDSA signature

Note: When an operation not listed here is written by the application, the write to CCOP[7:0] is ignored. All these operations implement protections against side-channel attacks. Write to CCOP is ignored when CCB_BUSY is set in CCB_SR.

32.7.2 CCB status register (CCB_SR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is reset by a system reset, or after clearing IPRST in CCB_CR. The CCB reset input from the RCC has no effect on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TAMP_EVT5	TAMP_EVT4	TAMP_EVT3	TAMP_EVT2	TAMP_EVT1	TAMP_EVT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCB_BUSY
		r	r	r	r	r	r								r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OPERR[5:0]						Res.	Res.	Res.	OPSTEP[4:0]				
		r	r	r	r	r	r				r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **TAMP_EVTi**: Tamper i flag (i = 5 to 0)

0: No event

1: Tamper event i has been raised. Global tamper event is raised. See *Section 32.3* for details on the events.

This bit is cleared by writing 0 to IPRST bit in CCR_CR. When at least one TAMP_EVT bit is set, CCB sets IPRST bit.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **CCB_BUSY**: CCB busy

This flag is set when CCB is busy, or when OPSTEP[4:0] ≠ 0 following a CCB reset (hardware reset or use of IPRST in CCB_CR).

0: CCB idle

1: CCB busy or PKA RAM is being cleared following a peripheral reset.

When IPRST and CCB_BUSY are set, the SAES registers are not accessible (write ignored, read returns 0), and writes to EN or START bits in PKA_CR are ignored.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **OPERR[5:0]**: Operation error

When an error is detected during a protected PKA operation, the CCB raises IPRST in CCB_CR, and OPSTEP[4:0] is set to 0x1F. The bitfield is cleared by setting IPRST to 0.

0x0: No error detected during protected PKA operation (if applicable)

0x1: CCB RCC reset has been detected when OPSTEP > 0x0

0x2: SAES RCC reset has been detected when OPSTEP > 0x0

0x3: PKA RCC reset has been detected when OPSTEP > 0x0

0x4: RNG RCC reset has been detected when OPSTEP > 0x0

0x5: The RCC has gated the CCB clock when OPSTEP > 0x0.

0x6: The RCC has gated the SAES clock when OPSTEP > 0x0.

0x7: The RCC has gated the PKA clock when OPSTEP > 0x0.

0x8: The RCC has gated the RNG clock when OPSTEP > 0x0.

0x9: Random in SAES_IV1 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xA: Random in SAES_IV2 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xB: Random in SAES_IV3 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xC: When OPSTEP = 0x17 SAES GCM TAG comparison with CCB_REFTAG failed, or a register different from SAES_DOUT was read in the SAES.

0xD: KEYSIZE ≠ 256 bits in SAES_CR when the SAES is enabled, and OPSTEP > 0x0.

0xE: CHMOD ≠ GCM in SAES_CR when 0x1 < OPSTEP < 0x18, or the SAES is not enabled, or KEYVALID is cleared when GCMPH > 0x0.

0xF: KMOD different from wrapped key mode in SAES_CR when the SAES is enabled, and OPSTEP = 0x1, or 0x1A.

0x10: KEYSEL ≠ 0x1 or 0x4 in SAES_CR when the SAES is enabled, KEYVALID is set, and OPSTEP = 0x1.

0x11: GCMPH not correct (not incremented) in SAES_CR when OPSTEP > 0x1.

0x12: MODE in SAES_CR does not match the expected value (selected by CCOP in CCB_CR) when OPSTEP > 0x1.

0x13: BUSY = 1 in SAES_SR when OPSTEP= 0x18 or 0x19. SAES must not be used when PKA is computing a protected operation.

0x14: EN = 0 in PKA_CR when OPSTEP > 0x2 (except 0x12). In other words, PKA must be enabled in any OPSTEP except 0x1, 0x2, or 0x12.

0x15: INITOK = 0 in PKA_SR when OPSTEP > 0x2 (except 0x12).

0x16: Authorized by local firewall, but unexpected AHB error detected when OPSTEP > 0x0.

0x17: LMF = 1 in PKA_SR (only ECDSA verification available) when OPSTEP > 0x0.

0x18: CCB_REFTAGRx registers are equal to 0 when OPSTEP > 0x12.

0x1A: Security, privilege protections inconsistent (see *Section 5: Global TrustZone® controller (GTZC)*) for the peripherals SAES, PKA, RNG, and CCB when OPSTEP > 0x0. This error is also triggered when the protections change while OPSTEP > 0x0,

0x1B: MODE in PKA_CR does not match the expected value (selected by CCOP in CCB_CR) when OPSTEP > 0x0.

0x1C: OPSTEP end-step event not found at OPSTEP= 0x0 (SAES KEYVALID is set). Use CCB IPRST before starting a coupling and chaining operation.

0x1D: OPSTEP end-step event not found at OPSTEP = 0x1.

0x1E: OPSTEP end-step event not found at OPSTEP = 0x2.

0x1F: OPSTEP end-step event not found at OPSTEP = 0x3.

0x20: OPSTEP end-step event not found at OPSTEP = 0xA.

0x21: OPSTEP end-step event not found at OPSTEP = 0x12.

0x22: OPSTEP end-step event not found at OPSTEP = 0x13.

0x23: OPSTEP end-step event not found at OPSTEP = 0x17.

For error codes 0x1C to 0x23, details on end-step events are found in *Section 32.5*. Those errors are also triggered when values written in SAES_CR register do not correspond to the selected CCOP and the expectations in the current OPSTEP.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **OPSTEP[4:0]**: Operation step

This read-only information indicates the current step of the coupling and chaining operation (indicated by CCOP). See *Section 32.5* for details. .

This bitfield can be cleared only using IPRST in CCB_CR. OPSTEP[4:0] = 0x1F when the IPRST procedure is in progress.

32.7.3 CCB reference tag register (CCB_REFTAGRx)

Address offset: 0x010 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

Writes to this register are ignored when CCB_BUSY is set in the CCB_SR register. The register is cleared when IPRST is set and CCB_BUSY is cleared.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFTAG{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFTAG{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REFTAG{i + 32 * x}**: Reference tag bits {i + 32 * x} (i = 31 to 0)

The application must write in these registers, the reference GCM tag of the required operation blob (respectively ECC signature, ECC key, or RSA key). This enables the PKA protected operation (respectively ECC signature/public key computation, scalar multiplication, or modular exponentiation). Refer to *Section 32.5* for details.

Write to this register is ignored unless OPSTEP = 0x12 in CCB_SR.

32.7.4 CCB register map

Table 284. CCB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CCB_CR	IPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCOP[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0x004	CCB_SR	Res.	Res.	TAMP_EVT5	TAMP_EVT4	TAMP_EVT3	TAMP_EVT2	TAMP_EVT1	TAMP_EVT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCB_BUSY	Res.	Res.	OPERR[5:0]						Res.	Res.	Res.	OPSTEP[4:0]				
	Reset value			0	0	0	0	0	0								0			0	0	0	0	0	0				0	0	0	0	0
0x010 + 0x4 * x (x=0 to 3)	CCB_ REFTAGRx	REFTAG{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

33 True random number generator (RNG)

33.1 RNG introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is an NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP 800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

33.2 RNG main features

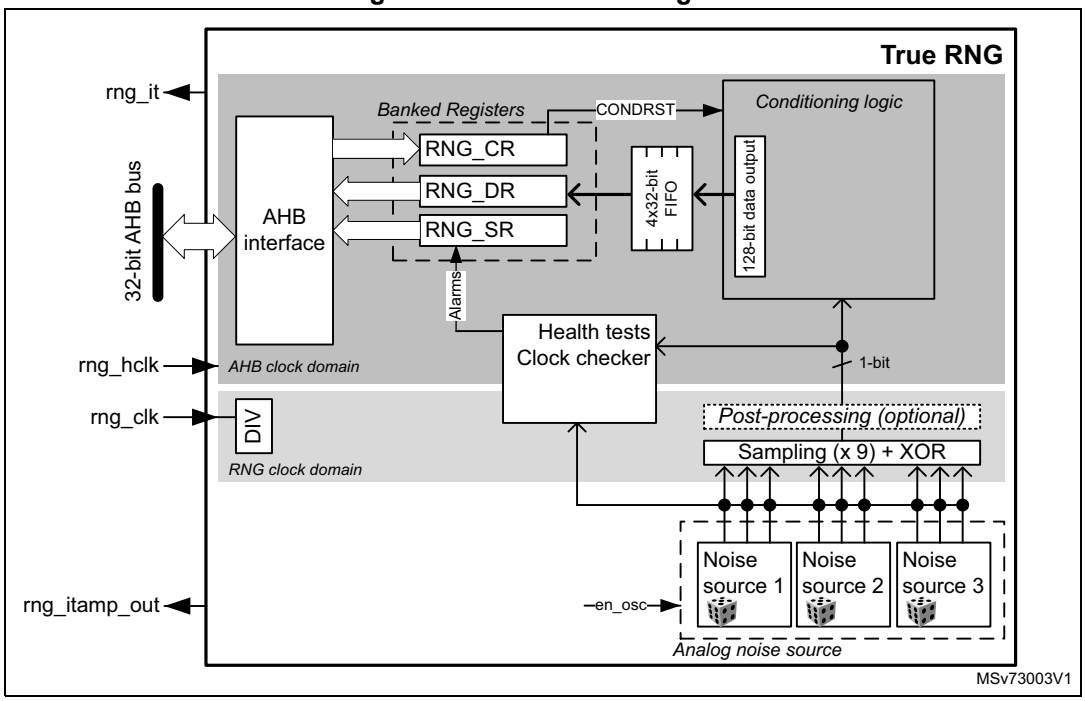
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by an NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a non-deterministic random bit generator (NDRBG).
- In the default configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- Embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- Can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- Has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

33.3 RNG functional description

33.3.1 RNG block diagram

Figure 303 shows the RNG block diagram.

Figure 303. RNG block diagram



33.3.2 RNG internal signals

Table 285 describes internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 285. RNG internal input/output signals

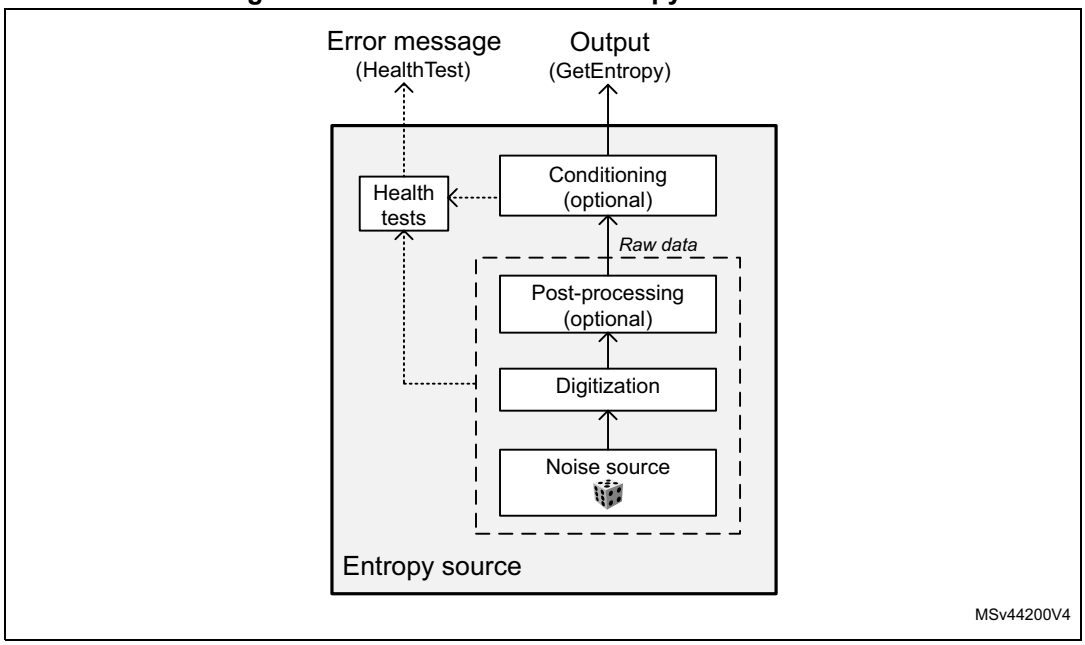
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	Digital output	RNG internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples. A reset and a new initialization are needed to use it again.

33.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary, it integrates all the required NIST components shown in *Figure 304*: an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block, and two interfaces used to interact with the entropy source (GetEntropy and HealthTest).

Figure 304. NIST SP800-90B entropy source model



Noise source

This component contains the nondeterministic, entropy-providing activity, ultimately responsible for the uncertainty associated with the bit string output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x3), each based on three XOR-ed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 33.3.8*. Multiple oscillators are also disabled for configuration A (see *Table 287*).
- The XOR-ing of all the noise sources into a single analog output.
- A sampling stage of this output is clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent from the AHB interface clock frequency (rng_hclk), with the possibility, for the software, to decrease the sampling frequency by using the integrated divider.

Note: The recommended clock frequencies and associated divider value are detailed in Section 33.6.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In the non-NIST configuration B (as defined in *Section 33.6.2*) a normalization debiasing is applied: half of the bits are taken from the sampled noise source, the other half from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bit string output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The time intervals required between two random number generations, and between the RNG initialization and availability of the first sample are detailed in *Section 33.5*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 33.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When the interrupts are enabled, an interrupt is generated when this flag transitions from 0 to 1. The interrupt is then cleared automatically by the RNG, as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts, then operates as expected, obtaining the assurance that failures are caught quickly, with high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the default RNG_HTCR0 configuration.

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 28 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 684 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source (XORed output and individual oscillator outputs, see *Figure 303: RNG block diagram*):
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous test:

  - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than the AHB clock cycle divided by 32.

4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in *Section 33.3.4*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 33.3.7*.

Note: An interrupt can be generated when an error is detected.

The health test thresholds are modified by changing the value in the corresponding RNG_HTCRx registers. See Section 33.6 for details.

33.3.4 RNG initialization

The RNG simplified state machine is shown in *Figure 305*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering the startup behavior test on the raw data samples and known-answer tests.
3. Start-up health tests are completed, with the conditioning stage ready to use.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 33.5*.

Figure 305. RNG initialization overview

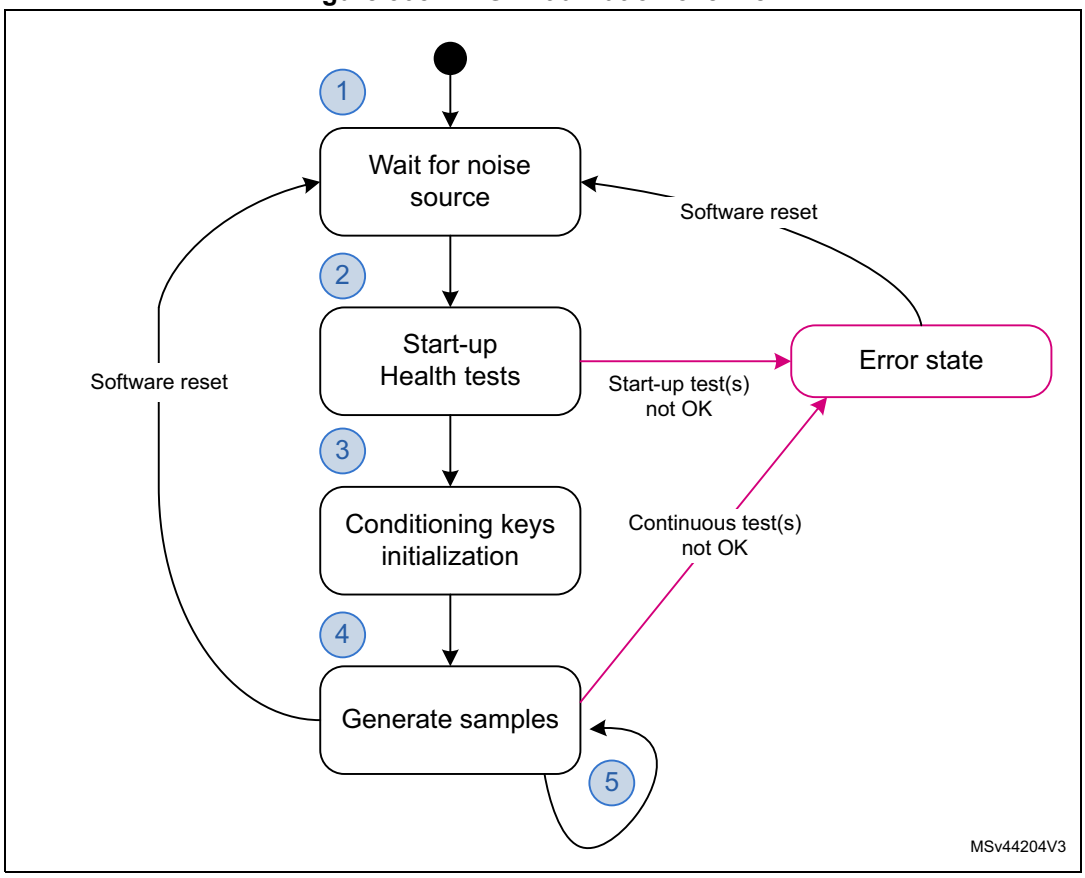


Figure 305 also highlights a possible software reset sequence, implemented by:

1. Write RNGEN = 0, then wait for BUSY = 0 in the RNG_SR register.
2. Write bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
3. Write RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
4. Wait for a random number to be ready, after initialization completes.

Note: When the RNG is reset by RCC (hardware reset), the configuration for optimal randomness is lost in the registers.

33.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check in *Section 33.6* if a specific RNG configuration is required for the application.
   – If this is the case, when bits RNGEN and BUSY=0, write in the RNG_CR register bit CONDRST = 1, together with the correct RNG configuration. Then perform a second write with bit CONDRST = 0, interrupt enable bit IE = 1, and RNG enable bit RNGEN = 1.

- If this is not the case, perform a write to the RNG_CR register with interrupt enable bit IE = 1 and RNG enable bit RNGEN = 1.

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 33.3.7* must be used.

To run the RNG in polling mode the following steps are recommended:

1. Check in *Section 33.6* if a specific RNG configuration is required for the application.
   - If this is the case, when bits RNGEN and BUSY = 0, write in the RNG_CR register bit CONDRST = 1, together with the correction RNG configuration. Then perform a second write to the RNG_CR register with bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If this is not the case, only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If the above conditions are true, read the content of the RNG_DR register up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, use the recovery sequence described in *Section 33.3.7*.

Note: When data are not ready (DRDY = 0) RNG_DR returns 0. It is recommended to verify that RNG_DR is different from 0. When this happens, a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required, it is possible to select a faster configuration by using the configuration B, described in *Section 33.6*. The gain in random number generation speed is summarized in *Section 33.5*.

Low-power operation

If power consumption is a concern, low-power strategies can be used, as described in *Section 33.3.8*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 33.3.3*.

33.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 33.6*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 33.3.1* for details (AHB and RNG clock domains).

33.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 33.3.6*), and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

When CECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 33.3.5*.

Note: CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred.

When SECS is set, values are available in the RNG_DR register, but it must not be used as described in *Section 33.3.5*.

To manage a seed error, follow the recommended sequence below.

It requires the CONFIGLOCK bit to be cleared.

- When ARDIS is cleared (auto-reset enabled):
  - Clear the SEIS bit in the RNG_SR register
  - If SECS is cleared RNG can be used as usual. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case, executes following step 1)
- When ARDIS is set (auto-reset disabled) the following sequence must be used to recover from a seed error.

1. Read both RNG_HTSRx registers. Using the previous HTSR values saved in RAM, compute the accumulated number of errors per oscillator (can be RPERR or ADERR errors). In a *nsmr* variable, mask the oscillators with accumulated number of errors greater than 3. This variable is used in step 3). Save the updated HTSR values in RAM.
   - During the first iteration, initialize *nsmr* variable with the RNG_NSCR value.
2. If fewer than 6 oscillators are active in the *nsmr* variable, reinitialize it with the RNG_NSCR value.
   - When RNG configuration is Configuration B, and the third time you reach this loop, go to Step 3 changing the RNG configuration to Configuration C. Both configurations are defined in *Section 33.6.2*.
3. Write CONDRST at 1 in RNG_CR, then update RNG_NSMR with *nsmr* value. Refer to *Section 33.7.8* for details.
4. Write CONDRST at 0 to finalize the reset of the RNG conditioning logic.
5. Wait for the CONDRST bit to be cleared by hardware, or poll for BUSY bit= 0. Then clear the SEIS interrupt status bit in the RNG_SR register.
6. If SECS is still set in RNG_SR a new seed error occurred (unlikely). In this case go back to step 1). If SECS is cleared, restart the generation of random numbers by setting RNGEN in RNG_CR.

If application has set the CONFIGLOCK bit (not recommended), do the following:

- When ARDIS is cleared (auto-reset enabled) clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.
- When ARDIS is set (auto-reset disabled), the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware, or poll for BUSY = 0. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

RNG tamper errors

When an unexpected error is found by the RNG, an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral, using the central reset management or the global SoC reset. A proper initialization of the RNG is required again.

Note: Seed errors do not trigger tamper errors.

33.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while the RNGEN = 0, the following features are available to the software:

- If there are valid words in the output buffer, four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register, four additional random numbers can be still be read from the RNG_DR register. If it is not the case, RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode), and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 33.5*.

When disabling the RNG, the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when the RNGEN bit is set to 1, unless the application resets the conditioning logic using the CONDRST bit in the RNG_CR register.

When the application wants to gate the RNG kernel clock, the software must issue the following sequence: first, disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR, and finally, disable the RNG kernel clock using the RCC.

Before entering a power mode where RNG is deactivated (like Stop mode), the software must first disable the RNG by clearing the RNGEN bit in the RNG_CR register, then wait for the BUSY bit to be cleared in the RNG_SR.

Note: The power modes where RNG is deactivated (retained or not available) can be found in the PWR section.

33.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 33.3.7*
- Clock error, see *Section 33.3.7*

Dedicated interrupt enable control bits are available as shown in *Table 286*.

Table 286. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 33.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources globally by using the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

33.5 RNG processing time

In the recommended configuration described in *Table 287*, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

If f_{RNG}= 48 MHz, $f_{threshold}$ = 1.6 x f_{RNG} = 77 MHz. The value of N is defined in *Section 33.6*.

Note: When CLKDIV is different from 0, f_{RNG} must take into account the internal divider ratio.

33.6 RNG entropy source validation

33.6.1 Introduction

To assess the amount of entropy available from the RNG, the peripheral has been tested using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

33.6.2 Validation conditions

The RNG true random number generator has been tested in the following conditions:

- RNG clock rng_clk = 48 MHz, with CED bit cleared in RNG_CR.
- Configurations are described in *Table 287* and *Table 288*. Only configuration A can be certified NIST SP800-90B. Refer to *Table 289* to select the best configuration for the application.
- Configuration B or C can be used when configuration A is not present in AN4230, or is not flagged as certified.

Table 287. RNG configurations

Configuration	RNG_CR bits					Loop number (N)	RNG_HTCR0 register	RNG_NSCR register
	NISTC bit	RNG_CONFIG1 [7:0]	CLKDIV [3:0]	RNG_CONFIG2 [2:0]	RNG_CONFIG3 [3:0](1)			
A	Refer to *NIST compliant RNG configuration* table in AN4230 available from *www.st.com*. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on *https://csrc.nist.gov/projects/cryptographic-module-validation-program.*							
B	1	0x83	0x0	0x0	0xF	3	0x0000 AAC7(2)	Default
C	1	0x84	0x5(3)	0x0	0xF	4		Default

1. RNG_CONFIG1[3:0] = i defines the loop number N. If i = 0, N =1. If NISTC = 0, N = 2 regardless of i.
2. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).
3. The noise source sampling must be 48 MHz or less. If the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 33.7.1* for details.

Table 288. Additional health test configurations

Configuration	RNG_HTCR1 register (oscillator set 1)	RNG_HTCR2 register (oscillator set 2)	RNG_HTCR3 register (oscillator set 3)
A	Refer to NIST compliant RNG configuration table in AN4230 available from *www.st.com*.		
B and C	Default	Default	Default

Table 289. Configuration selection

Section criteria	Configuration A	Configuration B	Configuration C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Not certified	
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using the AIS-31 test suite (T0 to T8).
2. When the speed is not enough for the application, an NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*Introduction to random number generation validation using the NIST statistical test suite for STM32 MCUs and MPUs*", available on *www.st.com*.

33.7 RNG registers

The RNG is associated with a control register, a data register, and a status register.

33.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]			
rs	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG configuration lock

Set this bit to lock the RNG configuration. Since CONFIGLOCK prevents the implementation of robust seed error management described in *Section 33.3.7*, it is recommended not to use it.

0: Writes to the RNG_NSMR, RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSMR, RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_NSMR, RNG_NSCR, RNG_CR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that sets any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

Before setting this bit to 1 RNGEN must be set to 0, and BUSY flag must be cleared in RNG_SR.

When CONDRST is set to 0, wait for BUSY flag to be cleared in RNG_SR. CONDRST bit then reads as 0.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **RNG_CONFIG1[7:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 33.6*.

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration for 128-bit output, two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 33.6* for recommended configurations.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset is enabled

1: Auto-reset is disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 33.3.7*.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enabled.

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS flag is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

This bit enables/disables the true random number generator.

0: True random number generator disabled.

1: True random number generator enabled.

Clearing RNGEN powers off the analog noise sources and gates the logic clocked by the RNG clock.

When clearing this bit verify that BUSY flag is cleared in RNG_SR before updating this register again.

Bits 1:0 Reserved, must be kept at reset value.

33.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0	r		r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)

1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 4 **BUSY:** Busy

This flag indicates whether RNG is idle or busy.

0: Idle

1: Busy

RNG is flagged as busy when a conditioning soft reset (using CONDRST) is in progress. It is also flagged busy when RNG is activated by application setting RNGEN bit, or when a connected peripheral fetches randoms via the dedicated RNG hardware bus.

More information on BUSY usage can also be found in *Section 33.3.8*.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

– Start-up or continuous repetition count test on noise source failed.

– Start-up or continuous adaptive proportion test on noise source failed.

– Start-up post-processing/conditioning sanity check failed.

After enabling the RNG, SECS bit clears once startup health tests completes successfully. If SECS does not clear, run the software reset sequence with CONDRST described in *Section 33.3.4*.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

This flag is raised to indicate valid (non-0) random data available in the RNG_DR register. It is cleared by hardware upon reading the RNG_DR register (which empties the RNG buffer). The DRDY flag can also be raised upon disabling RNG (upon clearing the RNGEN bit).

0: No valid random data ready.

1: Valid random data ready.

Raising DRDY generates an interrupt if enabled with the IE bit of the RNG_CR register.

33.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

This register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, valid when DRDY = 1. When DRDY = 0, the RNDATA value is 0.

The 0 value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

33.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0000 01FF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:6 **EN_OSC3[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

33.7.5 RNG health test control register x (RNG_HTCRx)

Address offset: 0x010 + 0x4 * x (x=0 to 3)

Reset value: 0x0000 72AC, 0x0003 FFFF, 0x0003 FFFF, 0x0003 FFFF

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTCFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTCFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests (repetition count test, adaptive proportion test) at position x. See *Section 33.6* for the recommended value. With its default value, health tests have a very small probability of triggering any error at position x.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

33.7.6 RNG health test status register 0 (RNG_HTSR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RPERR9	RPERR8	RPERR7	RPERR6	RPERR5	RPERR4	RPERR3	RPERR2	RPERR1	RPERRX
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:1 **RPERR{i}:** Repetitive error for oscillator *i* (i=1 to 9)

This bit is set when the health test logic associated to the oscillator *i* finds a repetitive test error. The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error. Repetition count test is defined in *Health checks* section.

0: No repetitive test error found in oscillator *i*.
1: Repetitive test error found in oscillator *i*.

Bit 0 **RPERRX:** Repetitive error after the XOR

This bit is set when the health test logic finds a repetitive test error after the XOR clock tree (see *Figure 303*). The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error.

0: No repetitive test error found after the XOR clock tree.
1: Repetitive test error found after the XOR clock tree.

33.7.7 RNG health test status register 1 (RNG_HTSR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ADERR9	ADERR8	ADERR7	ADERR6	ADERR5	ADERR4	ADERR3	ADERR2	ADERR1	ADERRX
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:1 **ADERR{i}:** Adaptative error for oscillator *i* (i=1 to 9)

This bit is set when the health test logic associated to the oscillator *i* finds an adaptative test error. The bit is cleared when application clears SEIS bit in RNG_SR. If ARDIS is cleared auto-reset prevents RNG to capture more than one test error. Adaptive proportion test is defined in *Health checks* section.

0: No adaptative test error found in oscillator *i*.
1: Adaptative test error found in oscillator *i*.

Bit 0 **ADERRX:** Adaptative error after the XOR

This bit is set when the health test logic finds an adaptative test error after the XOR clock tree (see *Figure 303*). The bit is cleared when application clears SEIS bit in RNG_SR If ARDIS is cleared auto-reset prevents RNG to capture more than one test error.

0: No adaptative test error found after the XOR clock tree.
1: Adaptative test error found after the XOR clock tree.

33.7.8 RNG noise source mask register (RNG_NSMR)

Address offset: 0x030

Reset value: 0x0000 01FF

Writing to this register is ignored if CONFIGLOCK = 1. It is recommended to have at least 6 oscillators running when loop number N = 3 (or higher). See *Section 33.6.2* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOSC 9	MOSC 8	MOSC 7	MOSC 6	MOSC 5	MOSC 4	MOSC 3	MOSC 2	MOSC 1
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **MOSC{i}:** Mask oscillator i (i = 1 to 9)

Clear this bit to masks the oscillator *i*, and deactivate its related health tests. The bit has no effect if the oscillator *i* is deactivated with the bit RNG_NSCR[i].

Application can mask an oscillator when it is not contributing to the generation of entropy.

0: Oscillator *i* is masked, and related health tests are deactivated.
1: Oscillator *i* is not masked. Related health tests are running.

33.7.9 RNG register map

Table 290. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]				RNG_ CONFIG2 [2:0]			NISTC	RNG_ CONFIG3 [3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0			0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
	Reset value																										0	0	0		0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3 [2:0]			EN_OSC2 [2:0]			EN_OSC1 [2:0]		
	Reset value																								1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR0	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0
0x014	RNG_HTCR1	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x018	RNG_HTCR2	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x01F	RNG_HTCR3	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x020	RNG_HTSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPERR9	RPERR8	RPERR7	RPERR6	RPERR5	RPERR4	RPER3	RPERR2	RPERR1	RPERRX
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	RNG_HTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADERR9	ADERR8	ADERR7	ADERR6	ADERR5	ADERR4	ADERR3	ADERR2	ADERR1	ADERRX
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x030	RNG_NSMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOSC9	MOSC8	MOSC7	MOSC6	MOSC5	MOSC4	MOSC3	MOSC2	MOSC1
	Reset value																								1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

34 AES hardware accelerator (AES)

This section only applies to STM32G54x/56x products.

In this section, any references to SAES only apply to STM32G56x products. The STM32G53x/54x/55x products do not have the SAES peripheral.

34.1 AES introduction

The AES hardware accelerator (AES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

AES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits.

AES has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required).

34.2 AES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data

- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if AES needs to process another message with a higher priority, then resume the original message

34.3 AES implementation

The devices have one AES peripheral, implemented as per the following table. It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

Table 291. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

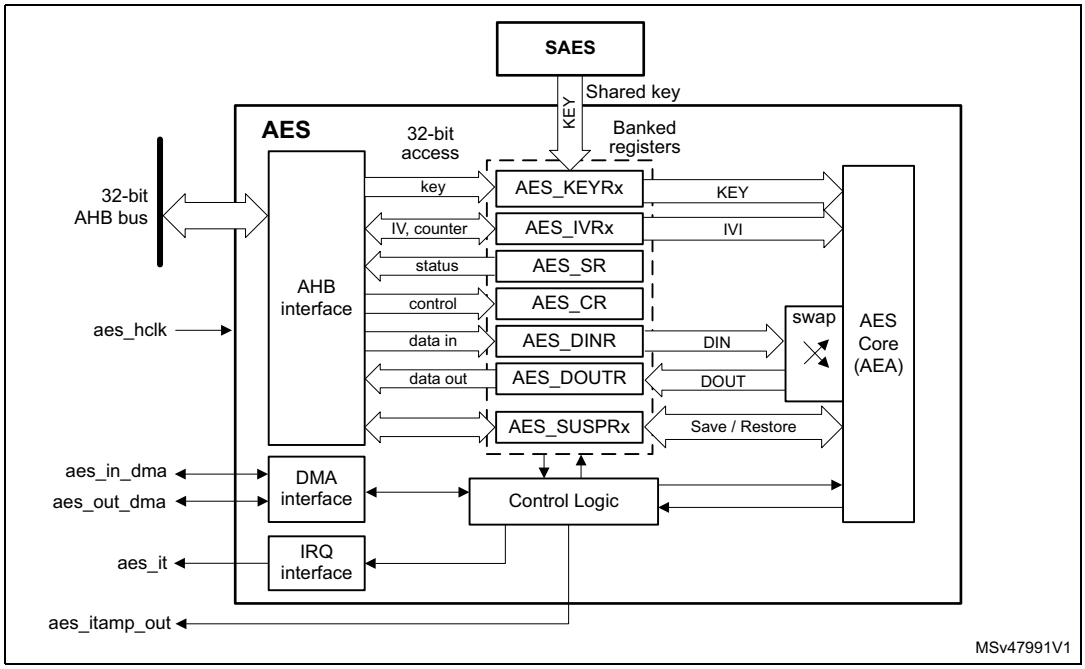
1. X = supported.

34.4 AES functional description

34.4.1 AES block diagram

Figure 306 shows the block diagram of AES.

Figure 306. AES block diagram



34.4.2 AES internal signals

Table 292 describes the user relevant internal signals interfacing the AES peripheral.

Table 292. AES internal input/output signals

Signal name	Signal type	Description
aes_hclk	Input	AHB bus clock
aes_it	Output	AES interrupt request
aes_in_dma	Input/Output	AES incoming data DMA single request/acknowledge
aes_out_dma	Input/Output	AES processed data DMA single request/acknowledge
aes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, AES automatically clears key registers. A reset is required for AES to be usable again.

34.4.3 AES reset and clocks

The AES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit controlled through the RCC.

34.4.4 AES symmetric cipher implementation

The AES hardware accelerator (AES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 293*. Those functions can be certified NIST PUB 140-3.

Table 293. AES approved symmetric key functions

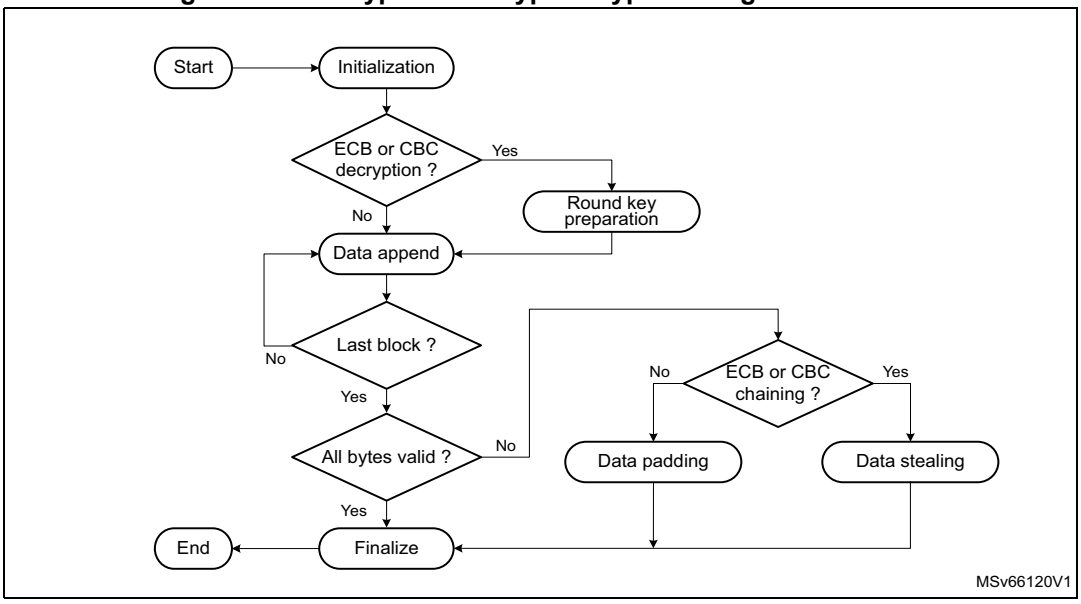
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

AES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any AES processing, following the sequence described in *Section 34.4.8*.

34.4.5 AES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 307. Encryption/ decryption typical usage



Initialization

The AES peripheral is initialized according to the chaining mode. Refer to *Section 34.4.9: AES basic chaining modes (ECB, CBC)* and *Section 34.4.10: AES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 34.4.7: AES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral when KEYVALID is set, by setting the EN bit of the AES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_ISR register, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).
3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the AES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES, by setting the CCFIE bit of the AES_IER register.
2. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the AES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the AES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the AES_DINR register and quit the interrupt service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and AES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in *Section 34.6: AES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for AES to compute a correct tag.
3. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 34.4.8* is not supported.

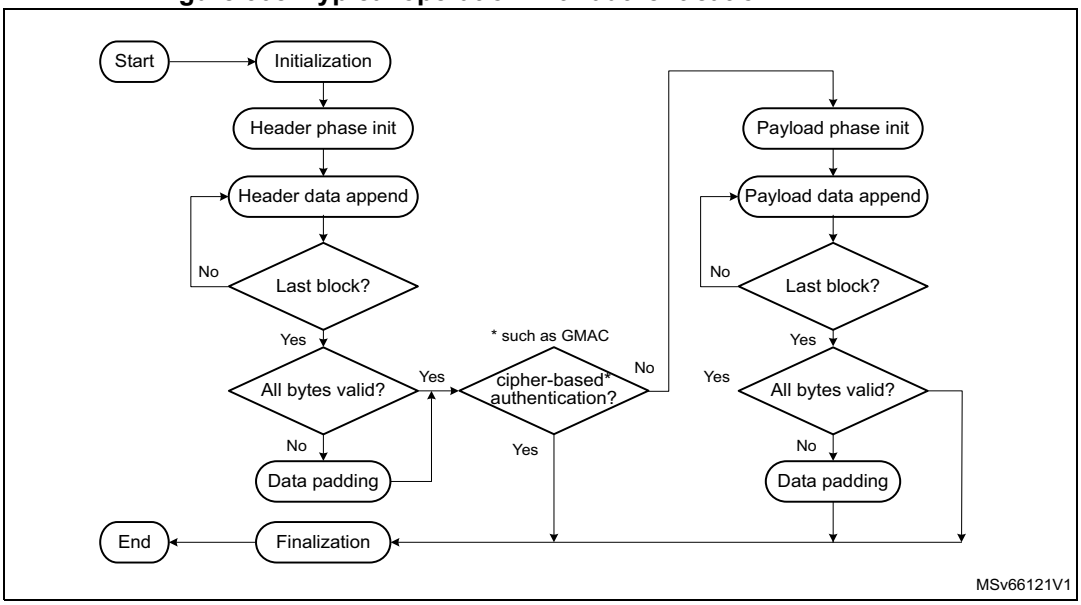
Note: The CCF flag has no use with this method because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

34.4.6 AES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 308. Typical operation with authentication



Section 34.4.11: AES Galois/counter mode (GCM) and *Section 34.4.13: AES counter with CBC-MAC (CCM)* describe detailed sequences supported by AES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by AES is described in *Section 34.4.12: AES Galois message authentication code (GMAC)*.

34.4.7 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since AES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since AES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

34.4.8 AES suspend and resume operations

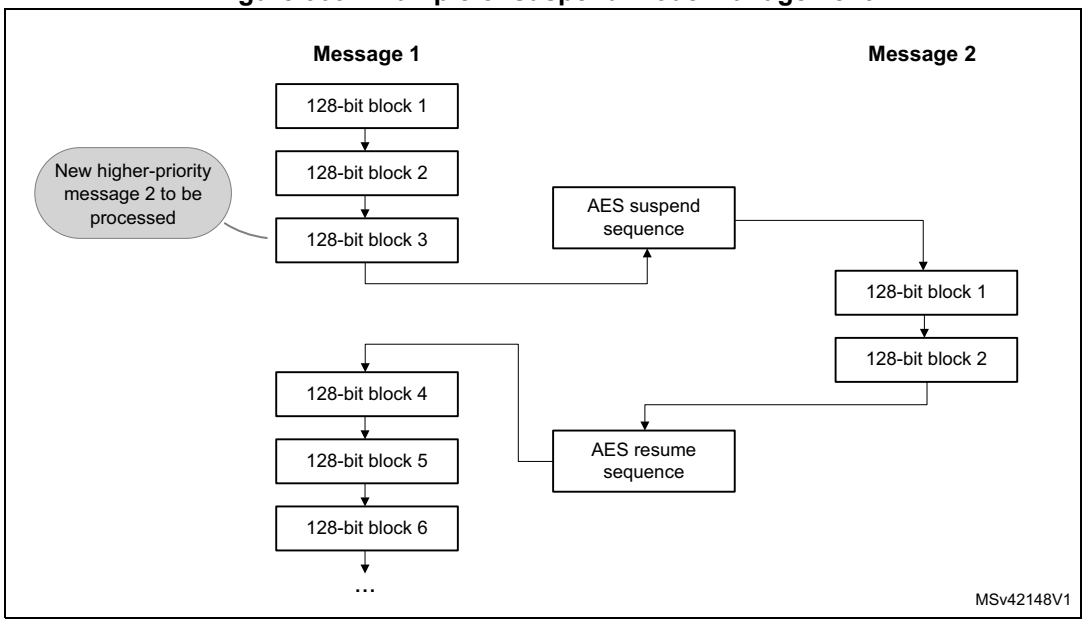
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as AES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

Figure 309 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 309. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

34.4.9 AES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 310 and *Figure 311* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 310. ECB encryption

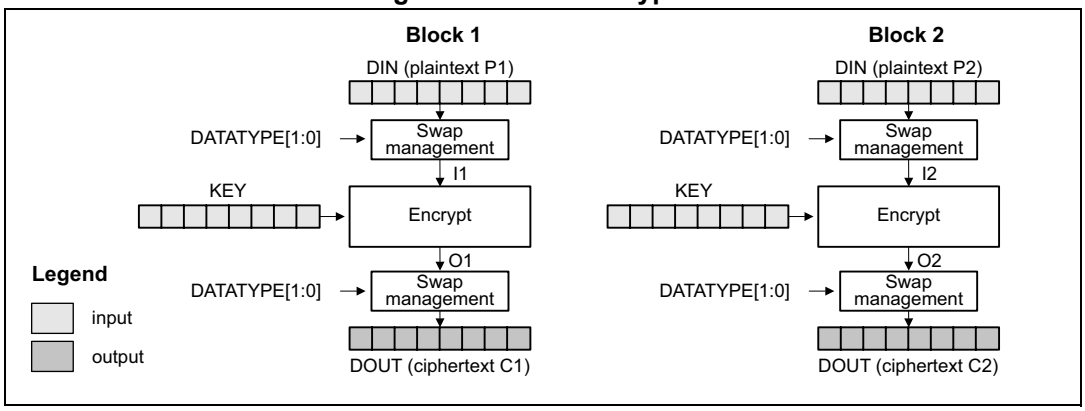
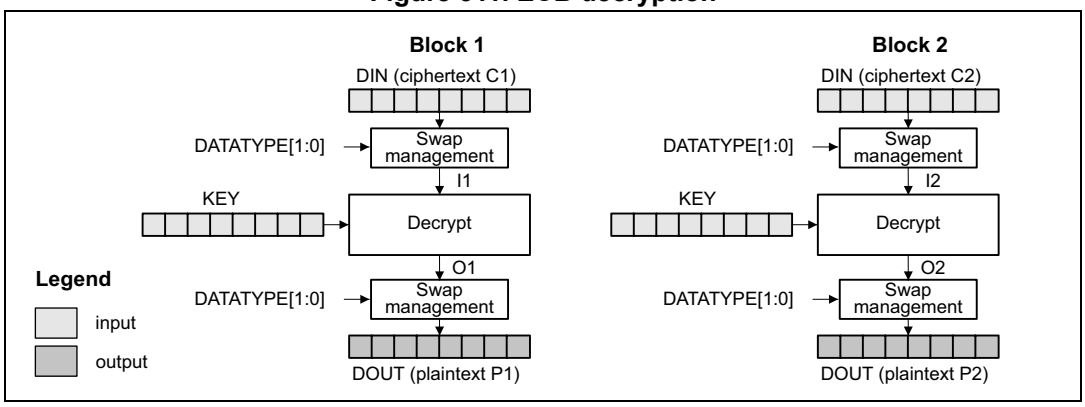


Figure 311. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 312 and *Figure 313* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 312. CBC encryption

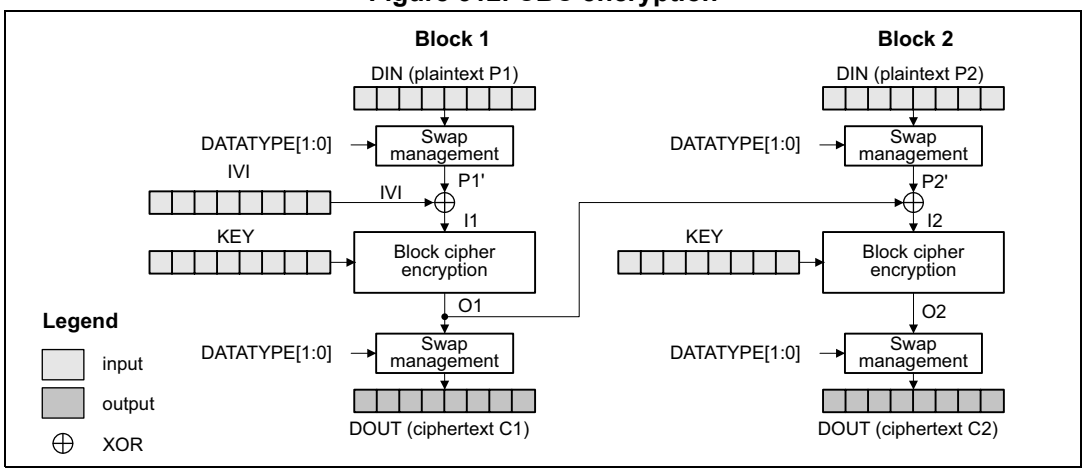
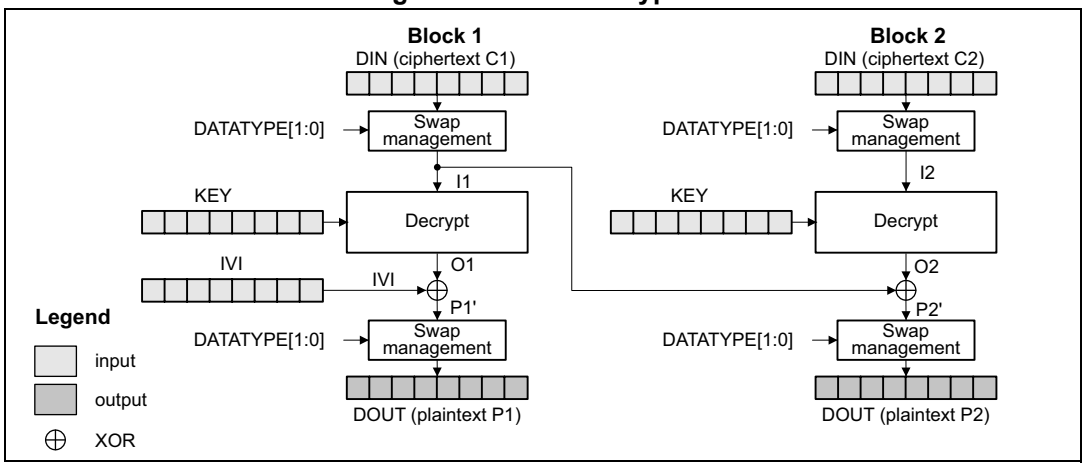


Figure 313. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 34.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 34.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 34.4.7*.
   b) Append the cleartext block into AES as described in *Section 34.4.5*, then read the AES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 34.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 34.4.14*).
4. Wait until KEYVALID is set (the key loading completed).
5. Enable the AES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
6. Wait until the CCF flag in the AES_ISR register is set.
7. Clear the CCF flag, by setting the CCF bit of the AES_ICR register. The decryption key is available in the AES core and AES is disabled automatically.
8. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
9. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
10. Enable the AES peripheral, by setting EN.
11. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 34.4.7*.
    b) Append the ciphertext block into AES as described in *Section 34.4.5*, then read the AES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
12. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. Read four times the AES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the AES_IVRx registers are altered during the data processing).

5. Disable the AES peripheral, by clearing EN.
6. Save the AES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Restore the AES_CR register (with correct KEYSIZE) then restore the AES_KEYRx registers. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 34.4.14*).
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
4. Restore the AES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the AES peripheral, by setting EN.

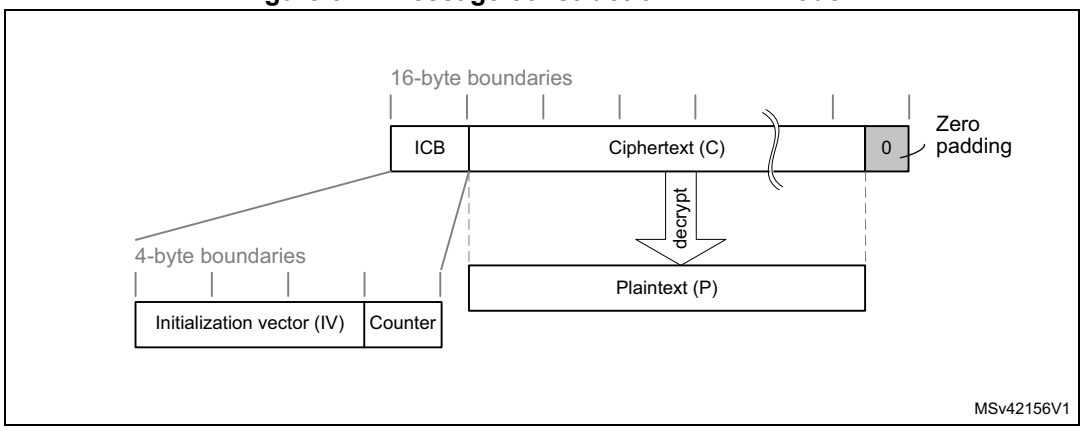
Note: It is not required to save the key registers as the application knows the original key.

34.4.10 AES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 314*.

Figure 314. Message construction in CTR mode



The structure of this message is:

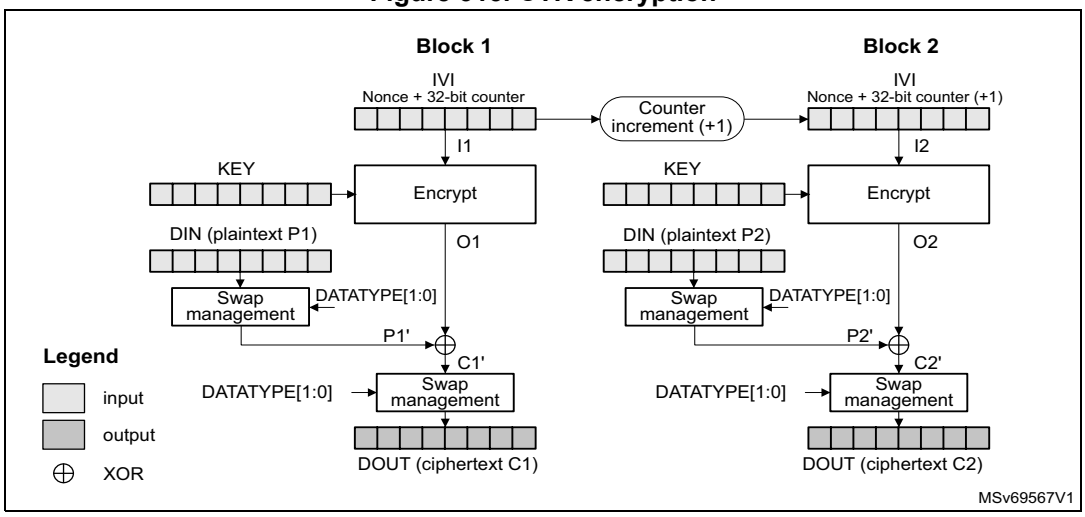
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 315 describes the counter (CTR) chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 315. CTR encryption



Initialization vectors in AES must be initialized as shown in *Table 294*.

Table 294. Counter mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 34.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers according to *Table 294*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 34.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.

7. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in *Section 34.4.5*, then read the AES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 34.4.9: AES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

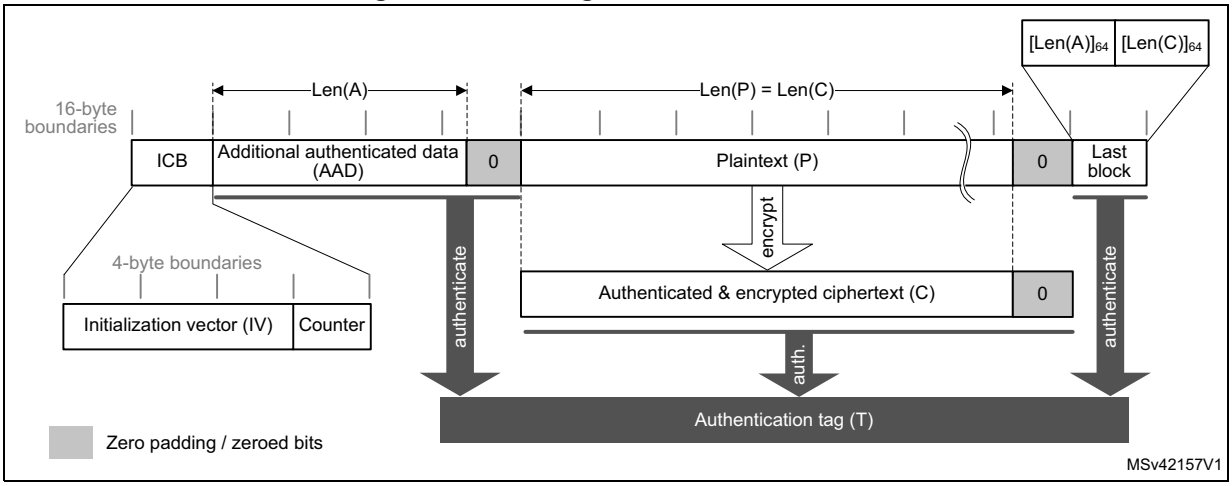
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

34.4.11 AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 316. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 296*.

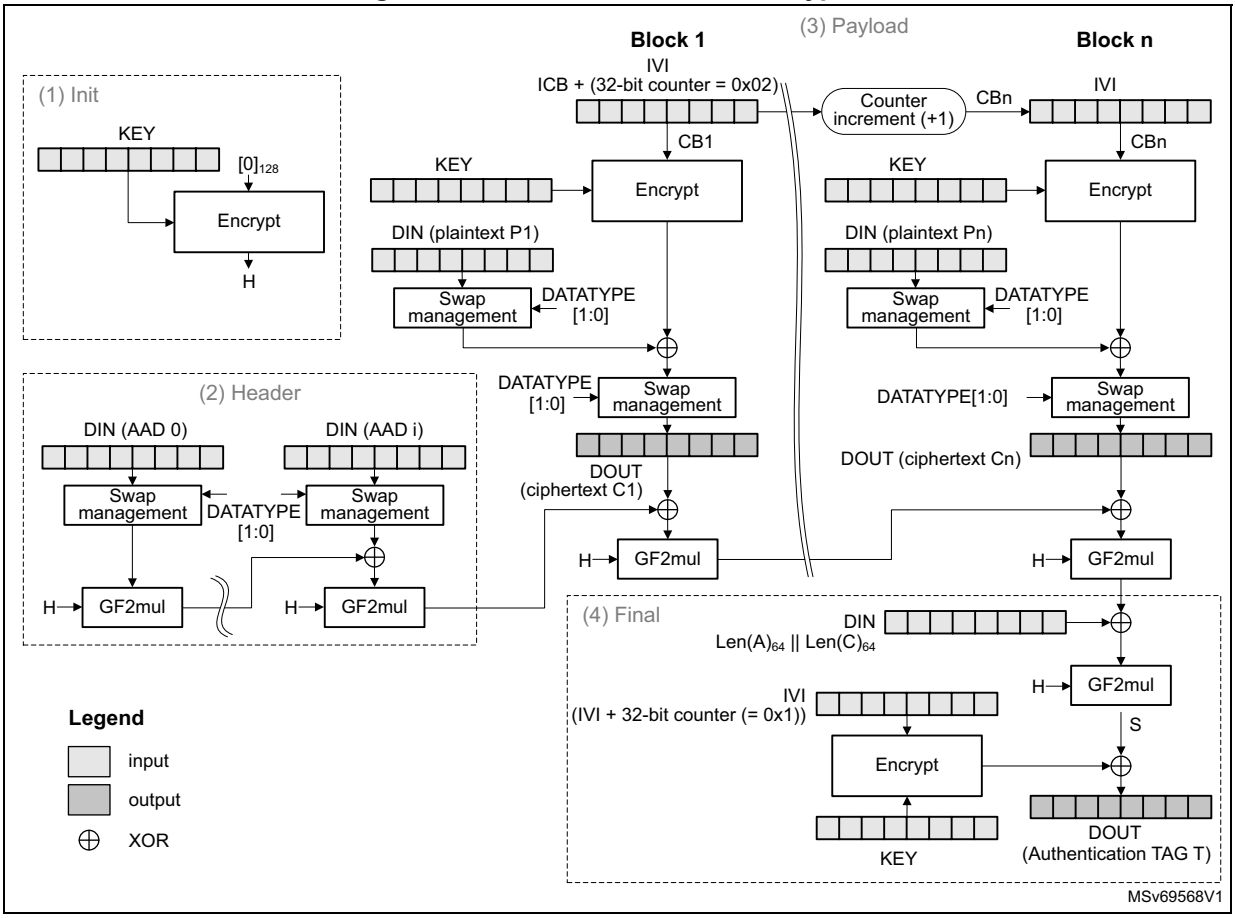
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 317 describes the GCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 317. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 295*.

Table 295. Initialization of IV registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 296*.

Table 296. GCM last block definition

Word order to AES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 34.4.6*, with the following sequence of events:

GCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the GCM initialization phase, by writing CPHASE[1:0] with 0x0.
3. Write the initialization vector in AES_IVRx registers according to *Table 295*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 34.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
7. Wait until the CCF flag is set in the AES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
8. Clear the CCF flag by setting the CCF bit of the AES_ICR register.

GCM header phase

9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to CPHASE[1:0]. Do not change the other configurations written during GCM initialization.
   b) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 34.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

11. Initialize payload phase:
    a) Select the GCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the AES peripheral by setting EN.

12. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 34.4.5*, then read the AES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

13. Select the GCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
14. Write the final GCM block into the AES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 296*.
15. Wait until the CCF flag in the AES_ISR register is set.
16. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
18. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag of the AES_ISR register, by setting the CCF bit of the AES_ICR register. When GCM encryption payload phase is selected, verify that BUSY is cleared in AES_SR, to ensure that GF2mul hash function is completed.
4. Save the AES_SUSPRx registers in the memory.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing EN.
7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

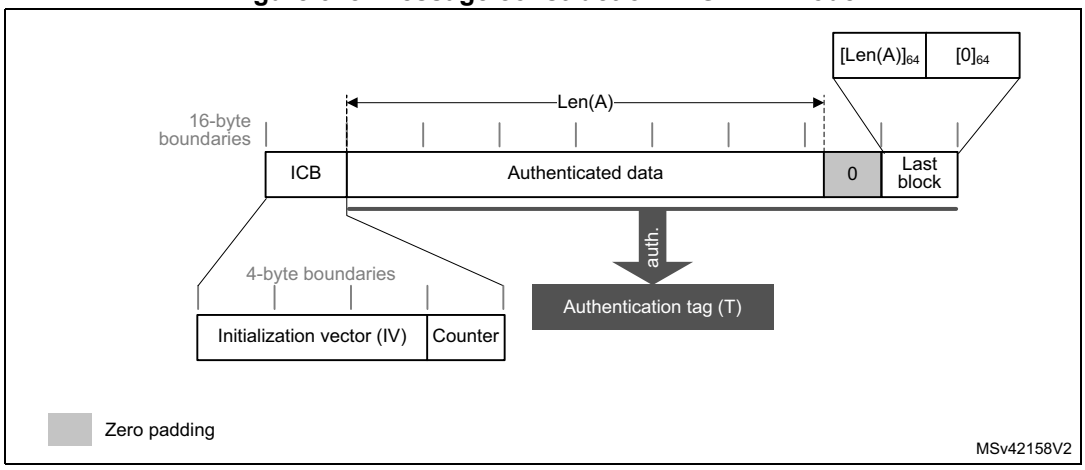
1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 34.4.14*).
5. Enable the AES peripheral, by setting EN.

34.4.12 AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

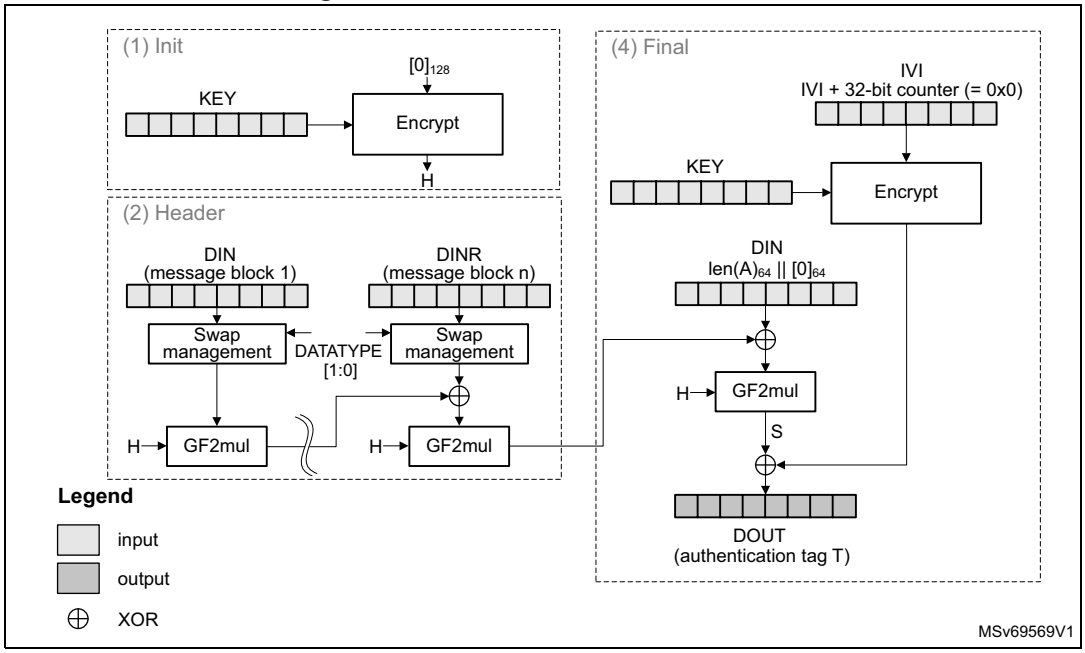
Figure 318. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 319 describes the GMAC chaining implementation in the AES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 319. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

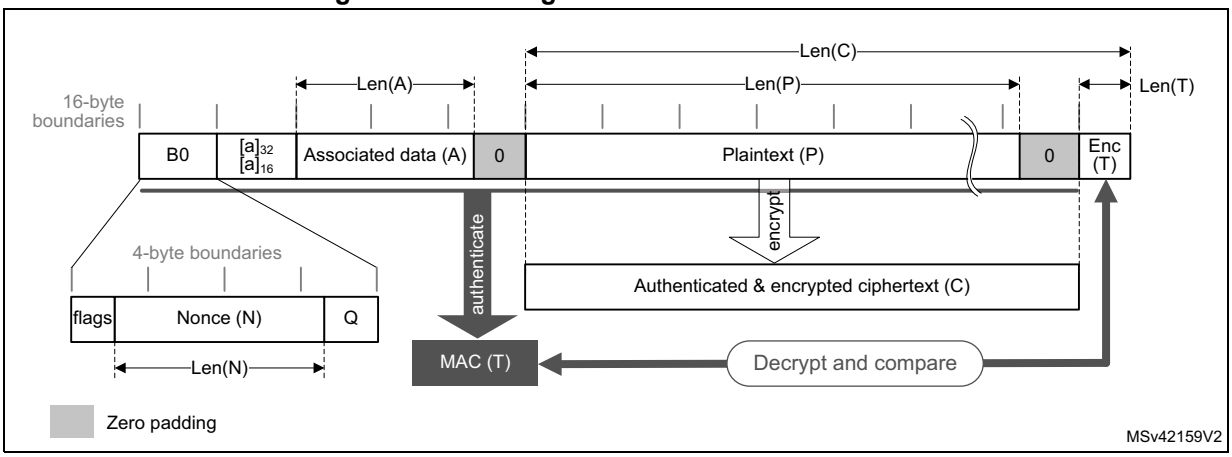
34.4.13 AES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 320. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 320*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 320*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

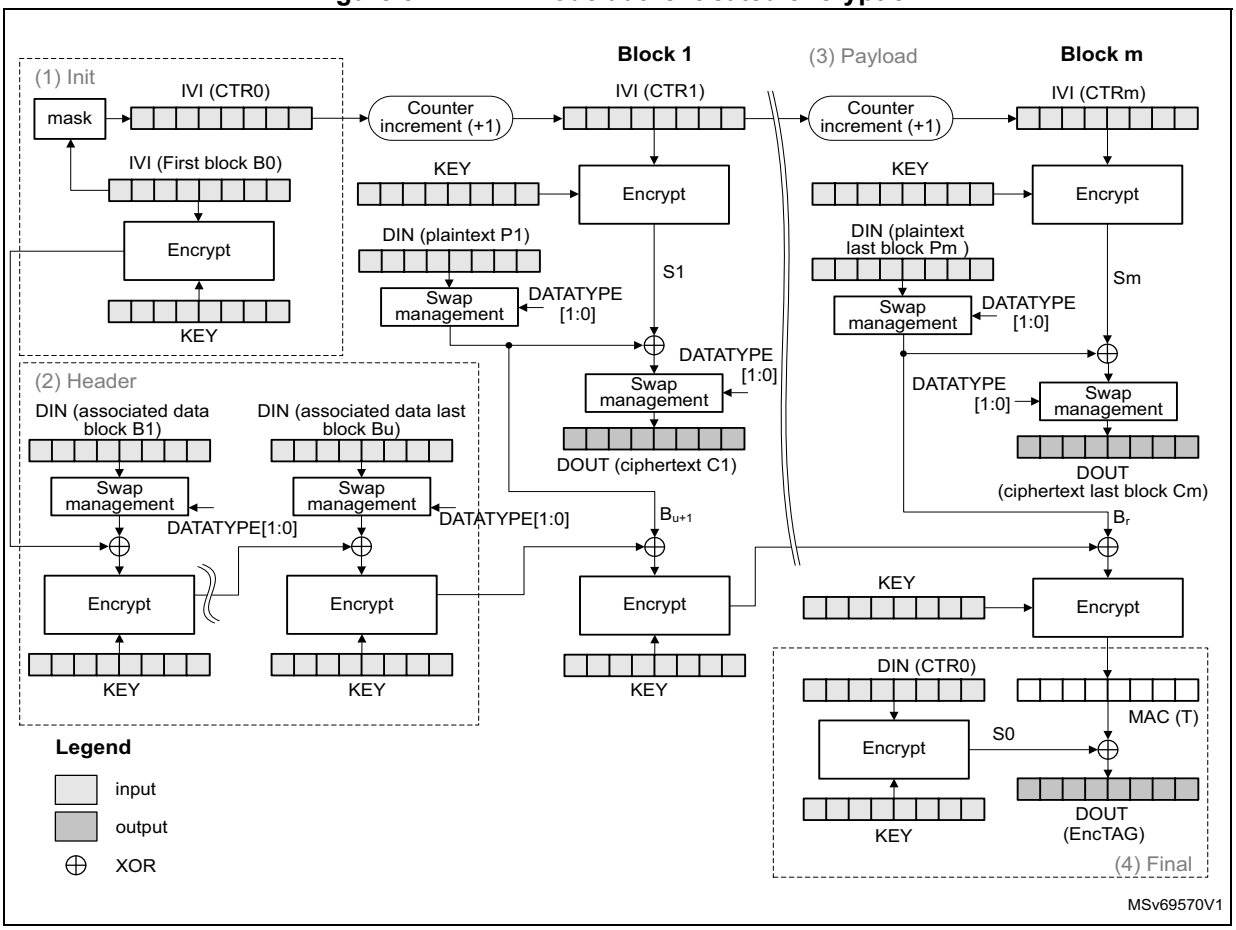
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 321 describes the CCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 321. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 297*.

Table 297. Initialization of IV registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

AES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 34.4.6*, with the following sequence of events:

CCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the CCM initialization phase, by writing CPHASE[1:0] with 0x0.
3. Write the B0 data in AES_IVRx registers according to *Table 297*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 34.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
7. Wait until the CCF flag in the AES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.

CCM header phase

9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the CCM header phase, by writing CPHASE[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
   c) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 34.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

11. Initialize payload phase:
    a) Select the CCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the AES peripheral, by setting EN.
12. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 34.4.5*, then read the AES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

13. Select the CCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
14. Wait until CCF flag in the AES_ISR register is set.
15. Get the CCM authentication tag, by reading the AES_DOUTR register four times.
16. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
17. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from AES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the AES_ISR register, by setting the CCF bit of the AES_ICR register.
4. Save the AES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the AES peripheral, by clearing EN.

7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. Restore AES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 34.4.14*).
5. Enable the AES peripheral, by setting EN.

34.4.14 AES key sharing with secure AES co-processor

The AES peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral and when it is ready, the AES application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in the section *SAES operations with shared keys* in the *SAES* section of this document.

1. In SAES peripheral, the application encrypts (wraps) the key to share in Shared-key mode (KMOD[1:0] at 0x2).
2. Each time the shared key is required in AES peripheral, the application decrypts it in the SAES peripheral in Shared-key mode (KMOD[1:0] at 0x2).
3. Once the shared key is decrypted (unwrapped) and loaded in SAES_KEYRx registers it can be shared with AES. To load the shared key in AES, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transfered by hardware into the AES_KEYRx registers and the BUSY flag in the AES_SR register set.
4. Once the key transfer is completed, the BUSY flag is cleared and the KEYVALID flag set in the AES_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KEIF flag is set in the AES_ISR register, either the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is completed, the AES is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in AES peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.

34.4.15 AES data registers and data swapping

Data input and output

A 16-byte data block enters the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the AES peripheral with four successive 32-bit word reads of the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 298* “no swapping” option for details.

Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both AES input and output.

Note: The data in AES key registers (AES_KEYRx) and initialization vector registers (AES_IVRx) are not sensitive to the swap mode selection.

The AES data swapping feature is summarized in *Table 298* and *Figure 322*.

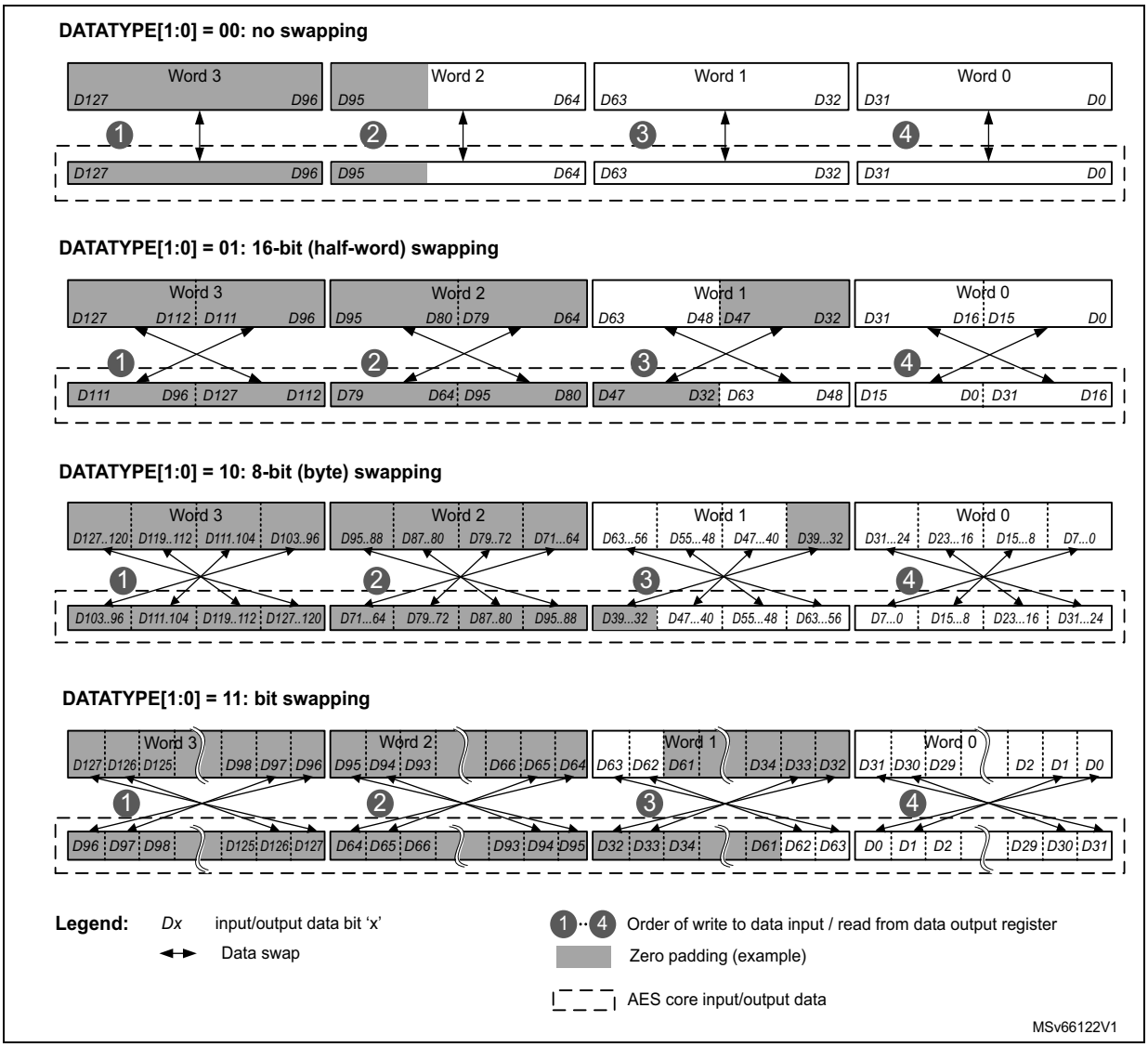
Table 298. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**

Table 298. AES data swapping example (continued)

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 322. 128-bit block construction according to the data type



Data padding

Figure 322 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

34.4.16 AES key registers

The eight AES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 299*. Reads are not allowed for security reason.

Note: In memory and in AES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 299. Key endianness in AES_KEYRx registers (128/256-bit keys)

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the AES_KEYRx registers are ignored when AES peripheral is enabled (EN bit set). The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes:

- AES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- AES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 34.4.18*).

34.4.17 AES initialization vector registers

The four AES_IVRx registers store the initialization vector (IV) information, as shown in *Table 300*. They can only be written if the AES peripheral is disabled (EN cleared).

Note: In memory and in AES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 300. IVI bitfield spread over AES_IVRx registers

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, AES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

34.4.18 AES error management

The AES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the AES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: AES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 34.5: AES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the AES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: AES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 34.5: AES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the AES_ISR register and clear the KEYVALID bit of the AES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 34.4.16: AES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.

- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 34.4.14: AES key sharing with secure AES co-processor* for details.

The KEIF flag is cleared with corresponding bit of the AES_ICR register. An interrupt is generated if the KEIE bit of the AES_IER register is set. For more details, refer to *Section 34.5: AES interrupts*.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring AES.

34.5 AES interrupts

There are multiple individual maskable interrupt sources generated by the AES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)

See *Section 34.4.18: AES error management* for details on AES errors.

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable AES interrupt sources individually by setting/clearing the corresponding enable bit of the AES_IER register.

The status of the individual maskable interrupt sources can be read from the AES_ISR register. They are cleared by setting the corresponding bit of the AES_ICR register.

Table 301 gives a summary of the available features.

Table 301. AES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)

1. Bit of the AES_ICR register.
2. Flag of the AES_SR register, mirrored by the flag RWEIF of the AES_ISR register.

34.6 AES DMA requests

The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the AES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

AES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with AES can be found in *Appending data using DMA* subsection of *Section 34.4.5: AES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into AES. AES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the AES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 34.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from AES. AES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the AES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, AES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when AES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

34.7 AES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 302. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles
128-bit	Encryption or decryption(1)	ECB, CBC, CTR	51
	Key preparation	-	59
256-bit	Encryption or decryption(1)	ECB, CBC, CTR	75
	Key preparation	-	82

1. Excluding key preparation time (ECB and CBC only).

Table 303. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Encryption/ Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Encryption/ Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

34.8 AES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

34.8.1 AES control register (AES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1:0]		NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
rw						rw	rw	rw	rw	rw	rw		rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: AES peripheral software reset

Setting the bit resets the AES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the AES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the AES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable.

0x2: Shared key mode. If shared key mode is properly initialized in SAES peripheral, the AES peripheral automatically loads its key registers with the data stored in the SAES key registers. The key value is available in AES key registers when BUSY bit is cleared and KEYVALID is set in the AES_SR register. Key error flag KEIF is set otherwise in the AES_ISR register.

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection
This bitfield defines the key length in bits of the key used by AES.
0: 128-bit
1: 256-bit
Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **CPHASE[1:0]:** Chaining phase selection
This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable
This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable
This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode
This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode
This bitfield selects the AES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 34.4.15: AES data registers and data swapping*.

0x0: No swapping (32-bit data).

0x1: Half-word swapping (16-bit data)

0x2: Byte swapping (8-bit data)

0x3: Bit-level swapping

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the AES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the AES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

– The bit cannot be set as long as KEYVALID is cleared

34.8.2 AES status register (AES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								r				r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in AES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KMOD[1:0] is different from zero, the BUSY flag is automatically set by AES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is usually set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the AES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 34.4.16: AES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether AES is idle or busy.
0: Idle
1: Busy
AES is flagged as idle when disabled (when EN is low) or when the last processing is completed.
AES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), or transferring a shared key from the SAES peripheral.
When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the AES_DINR register occurred. When set WRERRF bit has no impact on the AES operations.
0: No error
1: Unexpected write to AES_DINR register occurred during computation or data output phase.
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the AES_DOUTR register occurred. When set RDERRF bit has no impact on the AES operations.
0: No error
1: Unexpected read to AES_DOUTR register occurred during computation or data input phase.
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 0 Reserved, must be kept at reset value.

34.8.3 AES data input register (AES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.
Reads return zero.

34.8.4 AES data output register (AES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

34.8.5 AES key register 0 (AES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the AES_CR register.

Writes to AES_KEYRx registers are ignored when AES is enabled (EN bit set). When the key comes from the SAES peripheral (KMOD[1:0] at 0x2), writes to key registers are also ignored and they result in setting the KEIF bit of the AES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

34.8.6 AES key register 1 (AES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

34.8.7 AES key register 2 (AES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

34.8.8 AES key register 3 (AES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

34.8.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

AES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in AES_CR register

34.8.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

34.8.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

34.8.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

34.8.13 AES key register 4 (AES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

34.8.14 AES key register 5 (AES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

34.8.15 AES key register 6 (AES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

34.8.16 AES key register 7 (AES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

34.8.17 AES suspend registers (AES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSP[31:0]**: Suspend data

AES_SUSPRx registers contain the complete internal register states of the AES when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 34.4.8: AES suspend and resume operations* for more details.

Clearing EN bit of the AES_CR register clears this register to zero.

AES_SUSPRx registers are not used in other chaining modes than GCM, GMAC or CCM.

34.8.18 AES interrupt enable register (AES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

34.8.19 AES interrupt status register (AES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers.

0: No key error detected

1: Key information failed to load into key registers

The flag setting generates an interrupt if the KEIE bit of the AES_IER register is set. It also clears the key registers and the KEYVALID flag in the AES_SR register.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

KEIF is raised upon any of the following events:

–AES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 register, or reverse. For KEYSIZE set, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 then AES_KEYR4 then AES_KEYR5 then AES_KEYR6 then AES_KEYR7, or reverse).

–AES fails to load the key shared by the SAES peripheral (KMOD[1:0] = 0x2).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the AES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the AES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

34.8.20 AES interrupt clear register (AES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear
Setting this bit clears the KEIF status bit of the AES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear
Setting this bit clears the RWEIF status bit of the AES_ISR register, and clears both RDERRF and WRERRF flags in the AES_SR register.

Bit 0 **CCF:** Computation complete flag clear
Setting this bit clears the CCF status bit of the AES_ISR register.

34.8.21 AES register map

Table 304. AES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	AES_CR	IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1]	KMOD[0]	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0						0	0	0	0	0	0		0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 304. AES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	AES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	AES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 304. AES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	AES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
	Reset value																														0	0	0
0x304	AES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x308	AES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x309-0x3FF	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

35 Secure AES coprocessor (SAES)

This section only applies to STM32G56x products.

35.1 SAES introduction

The secure AES coprocessor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits, as well as special modes such as hardware secret key encryption/ decryption (wrapped-key mode) and key sharing with faster AES peripheral (shared-key mode).

SAES has the possibility to load by hardware STM32 hardware secret master keys (boot hardware key BHK and derived hardware unique key DHUK), usable but not readable by the application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the AES peripheral.

35.2 SAES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 480 or 680 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/ decryption (Wrapped-key mode)
- Using dedicated key bus, optional key sharing with faster AES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
  - Optional 128-bit or 256-bit hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together

- Security context enforcement for keys
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if SAES needs to process another message with a higher priority, then resume the original message

35.3 SAES implementation

The devices have one SAES peripheral, implemented as per the following table. It shares the key with the AES peripheral. For comparison, the AES peripheral is also included in the table.

Table 305. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

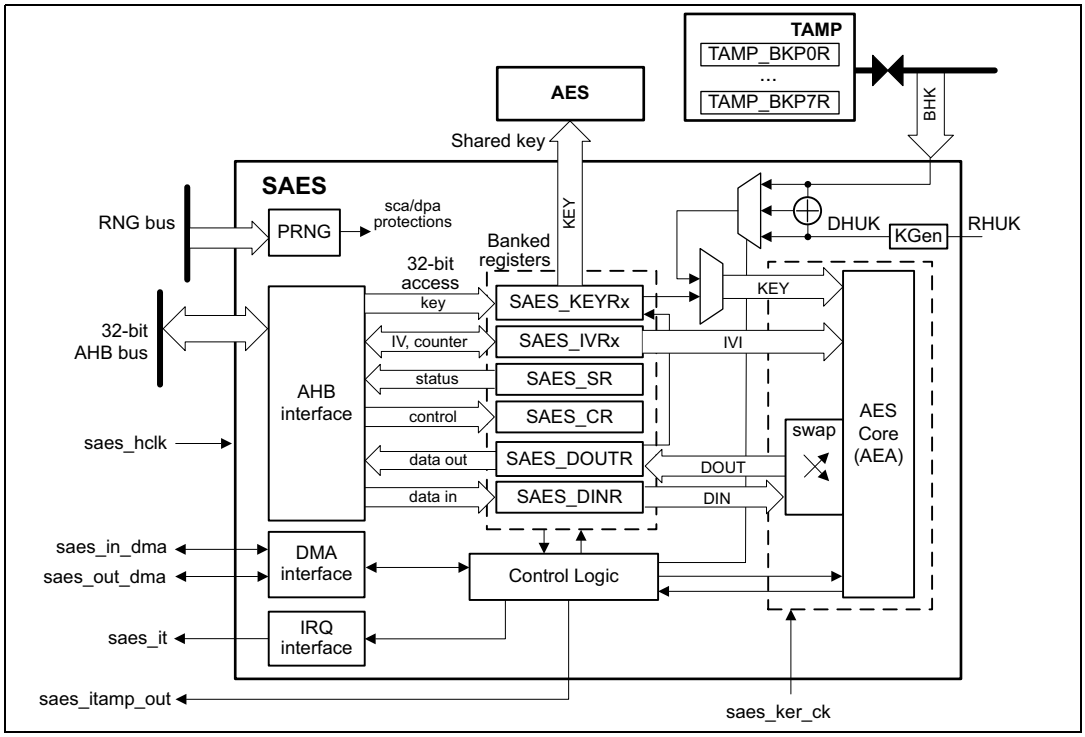
1. X = supported.

35.4 SAES functional description

35.4.1 SAES block diagram

Figure 323 shows the block diagram of SAES.

Figure 323. SAES block diagram



Note: AES represents the AES peripheral.

35.4.2 SAES internal signals

Table 306 describes the user relevant internal signals interfacing the SAES peripheral.

Table 306. SAES internal input/output signals

Signal name	Signal type	Description
saes_hclk	Input	AHB bus clock
saes_ker_ck	Input	SAES kernel clock.
saes_it	Output	SAES interrupt request
saes_in_dma	Input/Output	SAES incoming data DMA single request/acknowledge
saes_out_dma	Input/Output	SAES processed data DMA single request/acknowledge
saes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required for SAES to be usable again.
RHUK	Input	256-bit root hardware unique key (nonvolatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)
BHK(1)	Input	256-bit boot hardware key (BHK) stored in tamper-resistant secure backup registers and written by a secure code during boot. Once written, this key cannot be read nor written by any application until the next product reset.

1. Connected to a set of backup registers in TAMP peripheral that are written, then read/write locked, by the application software (see *Section 35.4.17* for details).

35.4.3 SAES reset and clocks

The SAES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit and a dedicated kernel clock, controlled through the RCC.

The SAES reset in RCC is mandatory when a hardware tamper event occurs in SAES. Refer to *Managing tamper errors* in *Section 35.4.19* for details.

After clocking then releasing the reset of the SAES peripheral in the RCC, SAES automatically draws random numbers from the RNG, setting the BUSY bit of the SAES_SR register. Refer to the *RNG error interrupt flag (RNGEIF)* in *Section 35.4.19* for details.

This background task must be completed before the device enters a low-power mode.

35.4.4 SAES symmetric cipher implementation

The secure AES coprocessor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 307*. Those functions can be certified NIST PUB 140-3.

Table 307. SAES approved symmetric key functions

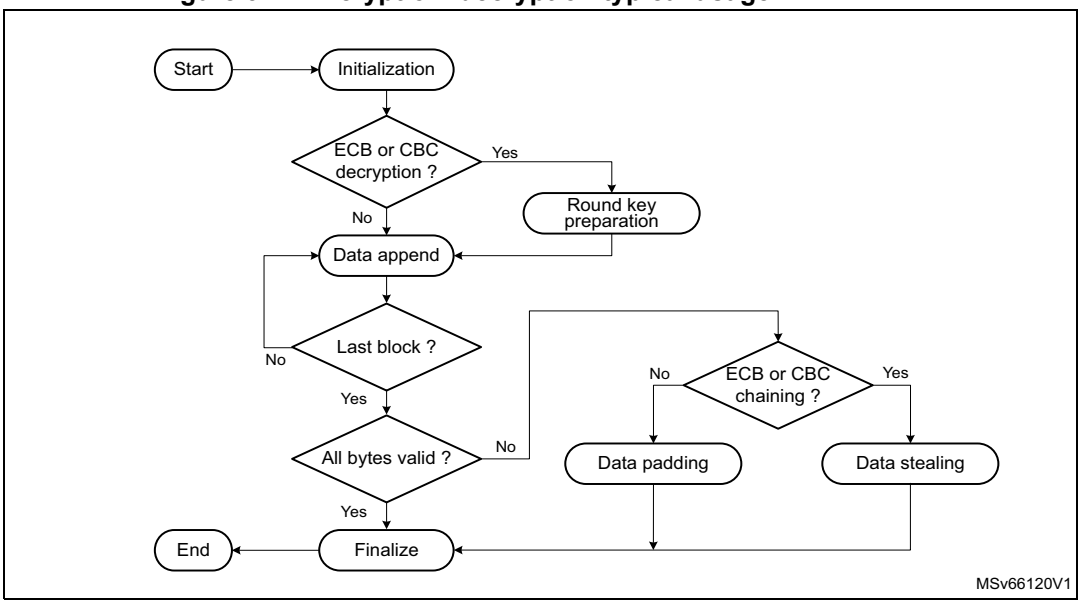
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

SAES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any SAES processing, following the sequence described in *Section 35.4.8*.

35.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 324. Encryption/ decryption typical usage



Initialization

The SAES peripheral is initialized according to the chaining mode. Refer to *Section 35.4.9: SAES basic chaining modes (ECB, CBC)* and *Section 35.4.10: SAES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 35.4.7: SAES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the SAES peripheral when KEYVALID is set, by setting the EN bit of the SAES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_ISR register, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using

NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the SAES_DINR register, to allow sending the key to the AES co-processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES, by setting the CCFIE bit of the SAES_IER register.
2. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the SAES_DINR register and quit the interrupt service routine.

Note: SAES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two SAES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in *Section 35.6: SAES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the

NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 35.4.8* is not supported.

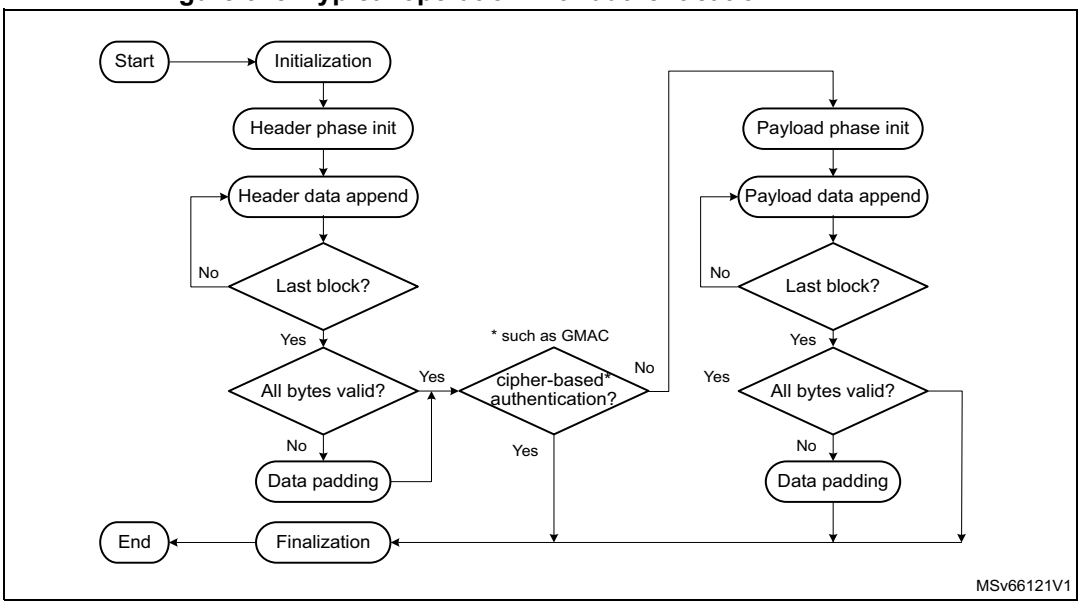
Note: The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

35.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 325. Typical operation with authentication



Section 35.4.11: SAES Galois/counter mode (GCM) and *Section 35.4.13: SAES counter with CBC-MAC (CCM)* describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by SAES is described in *Section 35.4.12: SAES Galois message authentication code (GMAC)*.

35.4.7 SAES ciphertext stealing and data padding

When using SAES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A*, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since SAES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since SAES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

35.4.8 SAES suspend and resume operations

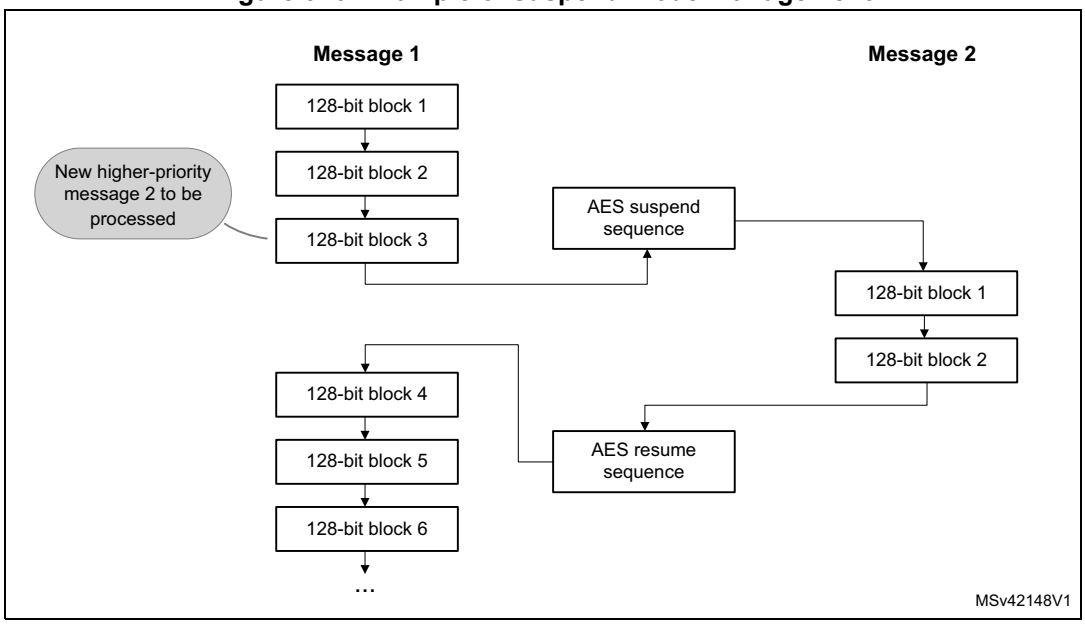
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

Figure 326 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 326. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

35.4.9 SAES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 327 and *Figure 328* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 327. ECB encryption

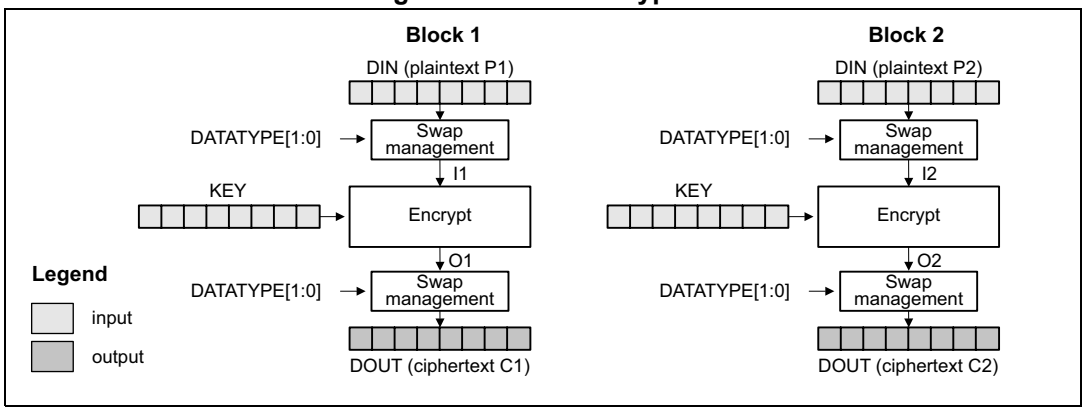
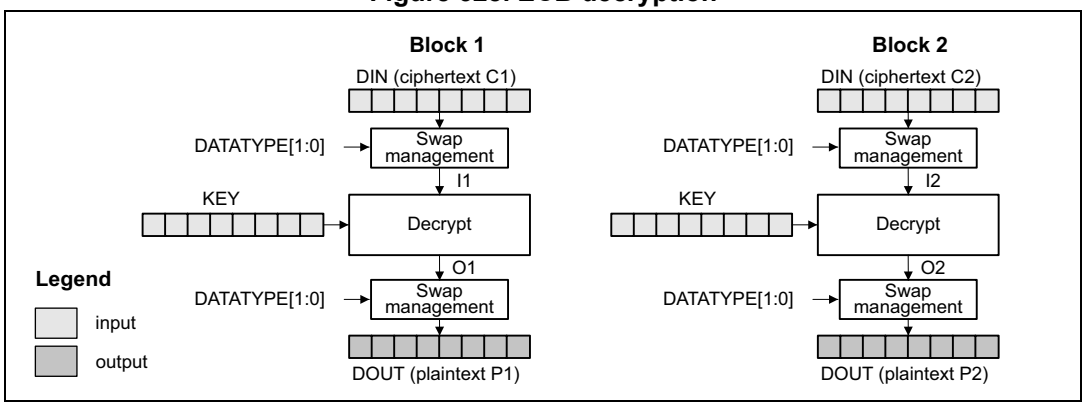


Figure 328. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 329 and *Figure 330* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 329. CBC encryption

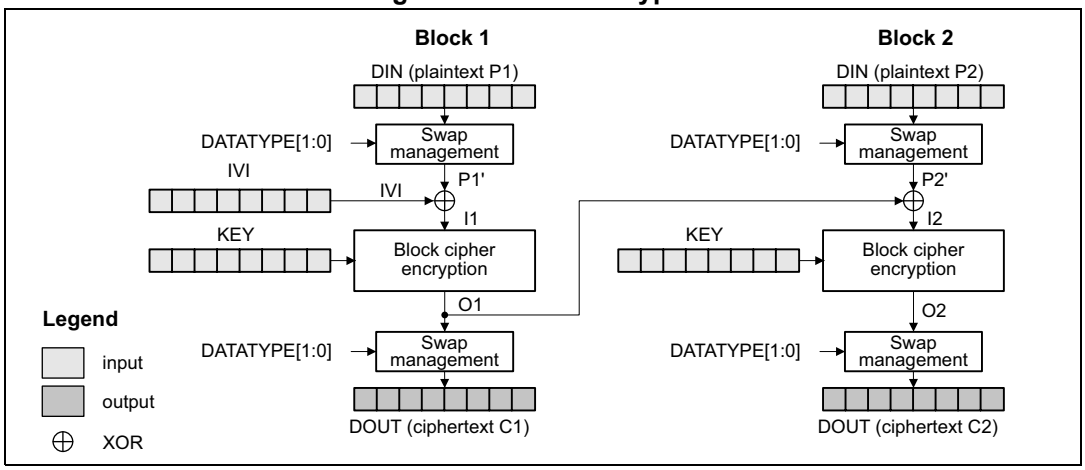
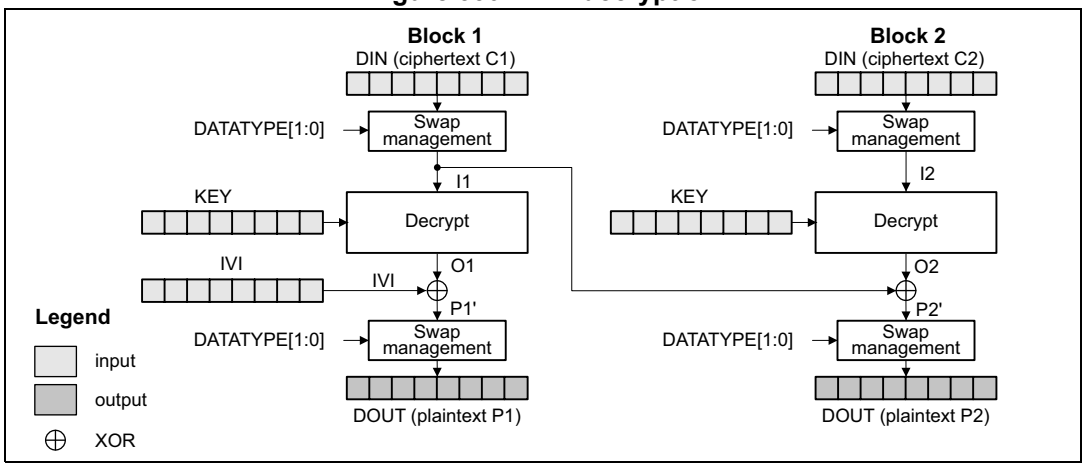


Figure 330. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 35.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 35.4.14* (wrapped keys) and *Section 35.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 35.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 35.4.7*.
   b) Append the cleartext block into SAES as described in *Section 35.4.5*, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 35.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 35.4.14* (wrapped keys) and *Section 35.4.15* (shared keys).

4. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 35.4.17: SAES key registers* for details.
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
7. Wait until the CCF flag in the SAES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.
9. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 35.4.7*.
    b) Append the ciphertext block into SAES as described in *Section 35.4.5*, then read the SAES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
13. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (KMOD[1:0] at 0x0).

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the SAES_ISR register is set (computation completed).
2. Read four times the SAES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the SAES_IVRx registers are altered during the data processing).
5. Disable the SAES peripheral, by clearing EN.
6. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Restore the SAES_CR register (with correct KEYSIZE) then restore the SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from key registers, refer to *Section 35.4.17: SAES key registers* for details.
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).

4. Restore the SAES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the SAES peripheral, by setting EN.

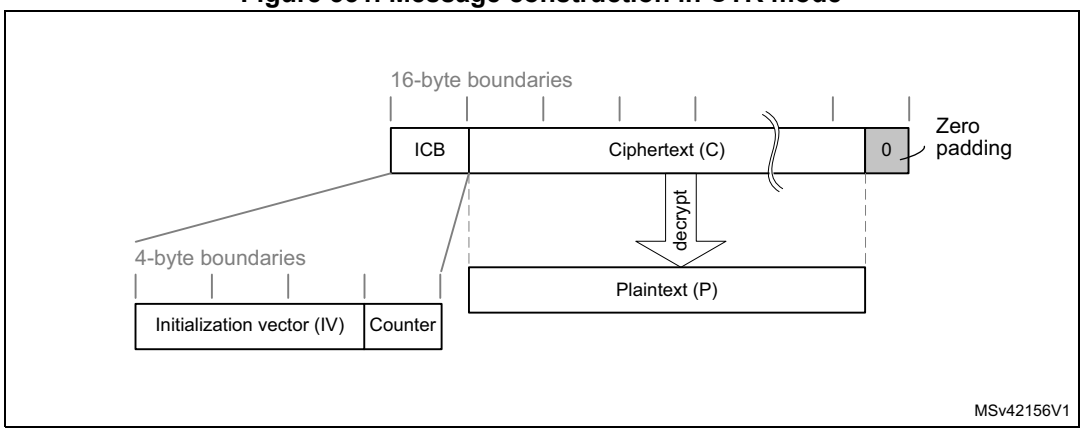
Note: It is not required to save the key registers as the application knows the original key.

35.4.10 SAES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 331*.

Figure 331. Message construction in CTR mode



The structure of this message is:

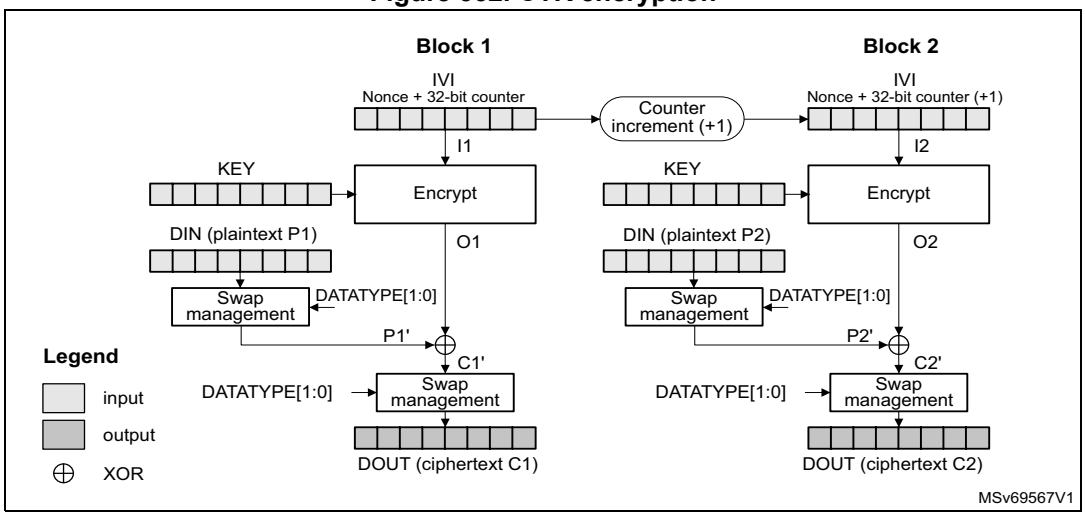
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 332 describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 332. CTR encryption



Initialization vectors in SAES must be initialized as shown in *Table 308*.

Table 308. Counter mode initialization vector definition

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 35.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 35.4.14* (wrapped keys) and *Section 35.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers according to *Table 308*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 35.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).

7. Enable the SAES peripheral, by setting EN.
8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in *Section 35.4.5*, then read the SAES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 35.4.9: SAES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

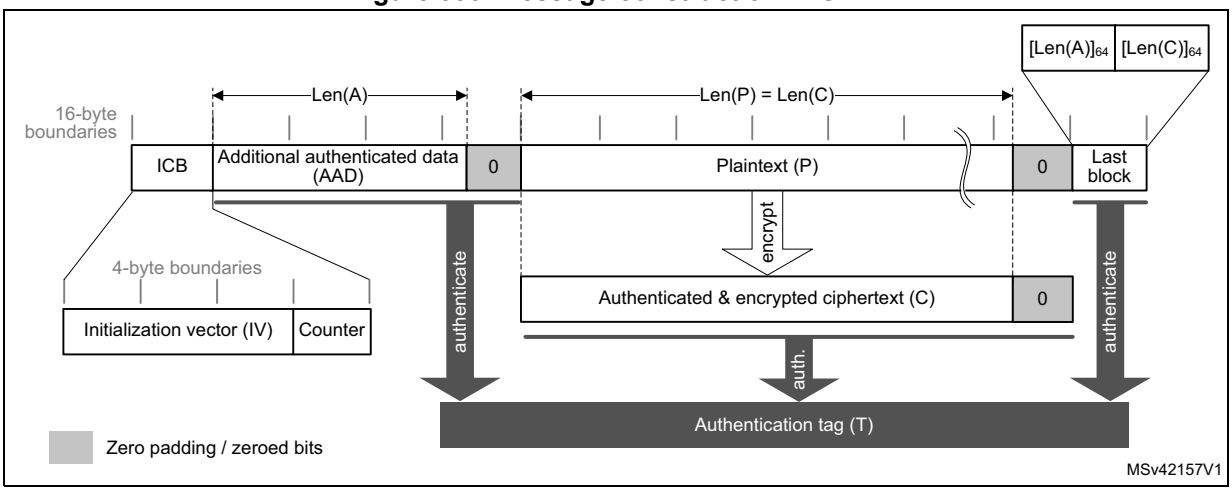
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

35.4.11 SAES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 333. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 310*.

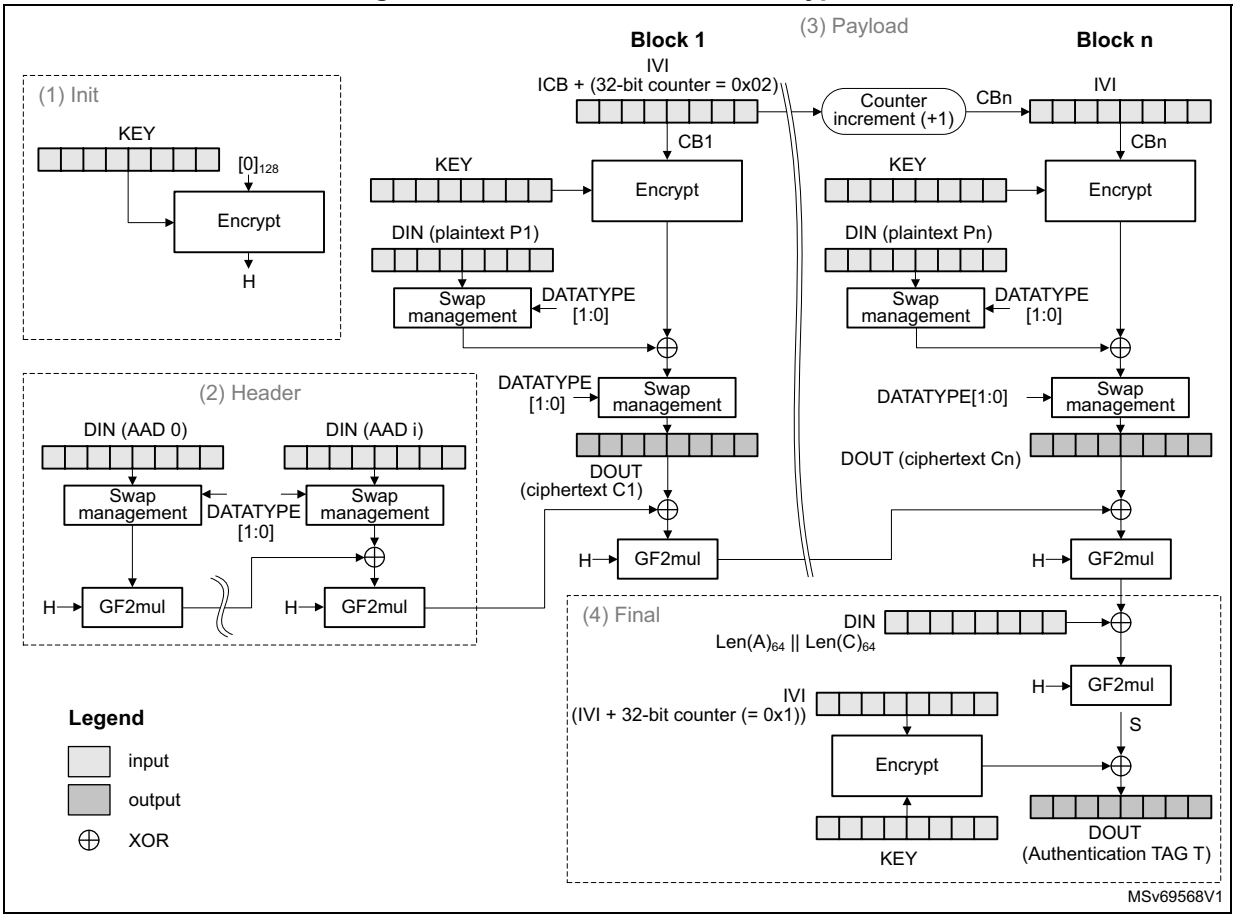
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 334 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 334. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 309*.

Table 309. Initialization of IV registers in GCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 310*.

Table 310. GCM last block definition

Word order to SAES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 35.4.6*, with the following sequence of events:

GCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (such as secure, nonsecure, specific CPU), also set KEYPROT.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 35.4.14* (wrapped keys) and *Section 35.4.15* (shared keys).
   - Select the GCM initialization phase, by writing CPHASE[1:0] with 0x0.
4. Write the initialization vector in SAES_IVRx registers according to *Table 309*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 35.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

GCM header phase

10. Initialize header phase:
    a) Select the GCM header phase, by writing 0x1 to CPHASE[1:0]. Do not change the other configurations written during GCM initialization.
    b) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 35.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Select the GCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral by setting EN.
13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 35.4.5*, then read the SAES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

14. Encryption only: wait until the BUSY flag in the SAES_SR register is cleared.
15. Select the GCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 310*.
17. Wait until the CCF flag in the SAES_ISR register is set.
18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.
19. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping is applied to tag data read from SAES_DOUTR.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

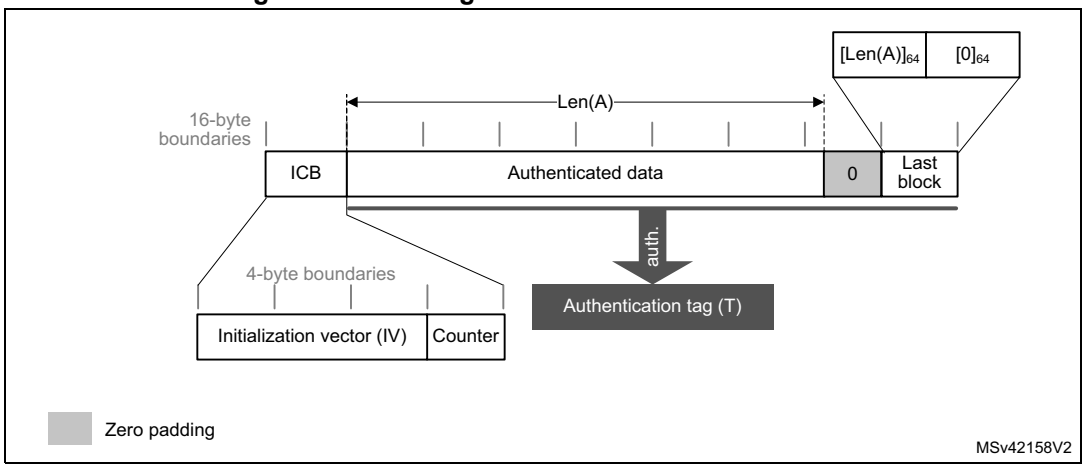
Suspend/resume operations are not supported in GCM mode.

35.4.12 SAES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

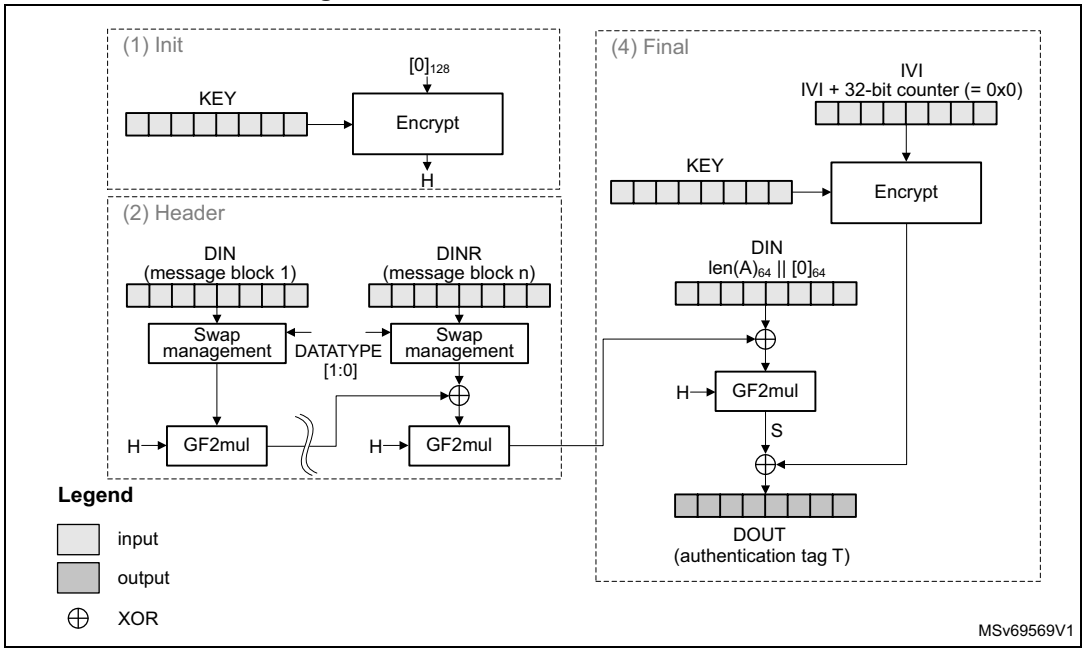
Figure 335. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 336 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 336. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

Suspend/resume operations are not supported in GMAC mode.

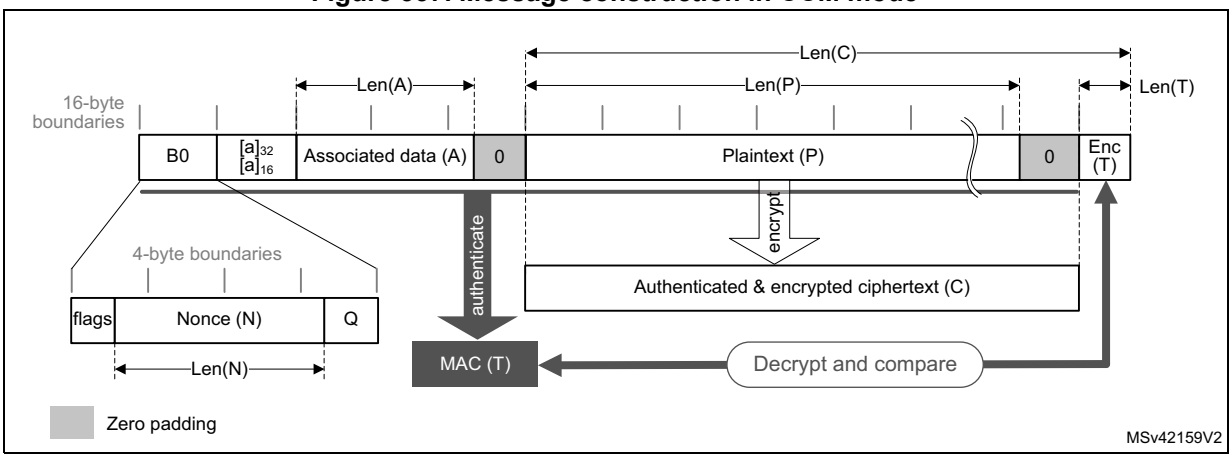
35.4.13 SAES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 337. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 337*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 337*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

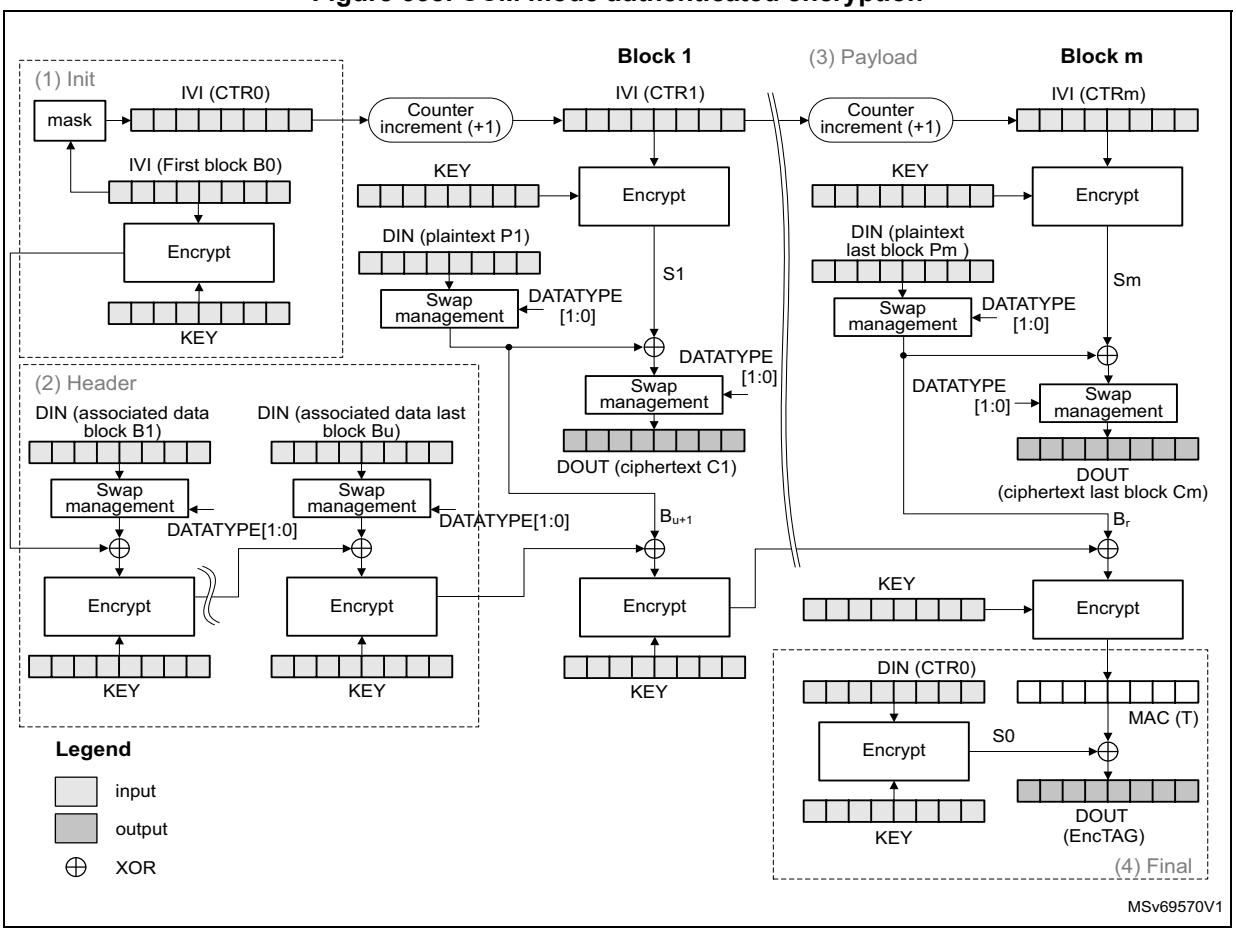
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C*, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 338 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 338. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 311*.

Table 311. Initialization of IV registers in CCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 35.4.6*, with the following sequence of events:

CCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context, also set the KEYPROT bit.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 35.4.14* (wrapped keys) and *Section 35.4.15* (shared keys).
   - Select the CCM initialization phase, by writing CPHASE[1:0] with 0x0.
4. Write the B0 data in SAES_IVRx registers according to *Table 311*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 35.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
8. Wait until the CCF flag in the SAES_ISR register is set.
9. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

CCM header phase

10. Initialize header phase:
    a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
    b) Select the CCM header phase, by writing CPHASE[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
    c) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 35.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Select the CCM payload phase, by writing CPHASE[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral, by setting EN.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 35.4.5*, then read the SAES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing CPHASE[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
15. Wait until CCF flag in the SAES_ISR register is set.
16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the SAES_ISR register is set (computation completed).
2. In the payload phase, read four times the SAES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.
4. Save the SAES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the SAES peripheral, by clearing EN.
7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
3. Restore SAES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from the key registers, refer to *Section 35.4.17: SAES key registers* for details.
5. Enable the SAES peripheral, by setting EN.

35.4.14 SAES operation with wrapped keys

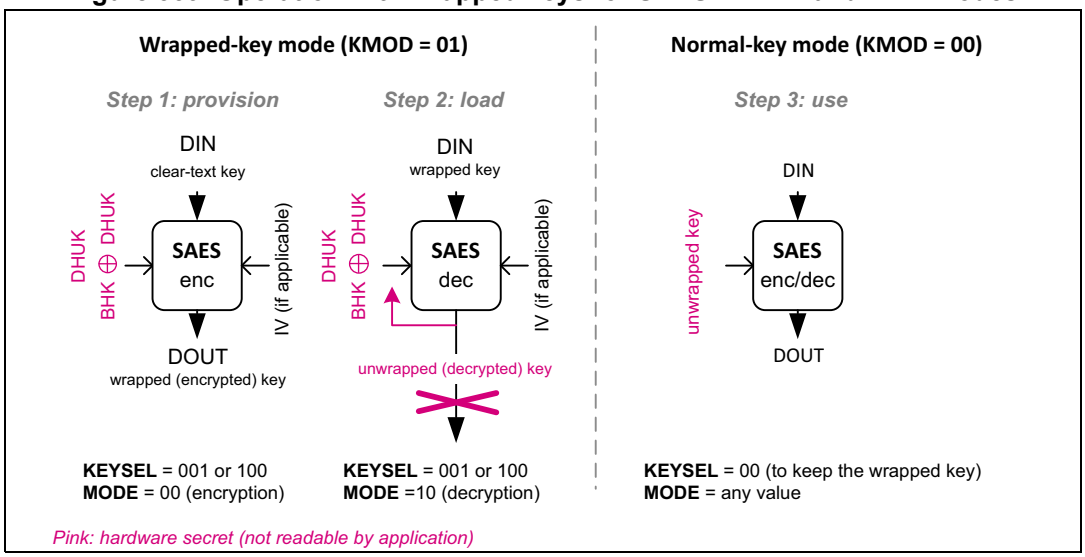
SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Operation with wrapped keys for SAES in ECB and CBC modes

Figure 339 summarizes how to wrap or unwrap keys for SAES in ECB and CBC modes. To protect the wrapped key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 339. Operation with wrapped keys for SAES in ECB and CBC modes



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

Key wrapping for SAES

The recommended sequence to wrap (that is, encrypt) a key is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (MODE[1:0] at 0x0)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. This information is used both for the encryption key and for the key to be encrypt.
   - Select wrapped key mode by writing KMOD[1:0] with 0x1
4. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 35.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (DHUK loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINR register four times to input the key to encrypt (MSB first, see *Table 313*).
9. Wait until CCF flag is set in the SAES_ISR register.
10. Get the encrypted key (MSB first) by reading the SAES_DOUTR register four times. Then clear the CCF flag, by setting the CCF bit in SAES_ICR register.
11. Repeat steps *8.* to *10.* if KEYSIZE is set.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected.

Key unwrapping for SAES

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the chaining mode used during the wrapping process (CHMOD[2:0] at 0x0 or 0x1) in *key derivation* mode (MODE[1:0] at 0x1)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size used during the wrapping process, with KEYSIZE. This information is used both for the decryption key and for the key to decrypt.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. With KEYSEL[2:0], select the same key source as when the key was wrapped/encrypted.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN bit in SAES_CR to enable the peripheral
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the *decryption* mode (MODE[1:0] at 0x2). Other parameters are unchanged.
10. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first, see *Table 313*).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
14. Repeat steps *12.* and *13.* if KEYSIZE is set.
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Decrypted wrapped key can be shared with an application running in a different security context (different security attribute) if KEYPROT bit was cleared during step 3.

Note: When KMOD[1:0] = 0x1 (wrapped key) and MODE[1:0] = 0x2 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERRF).

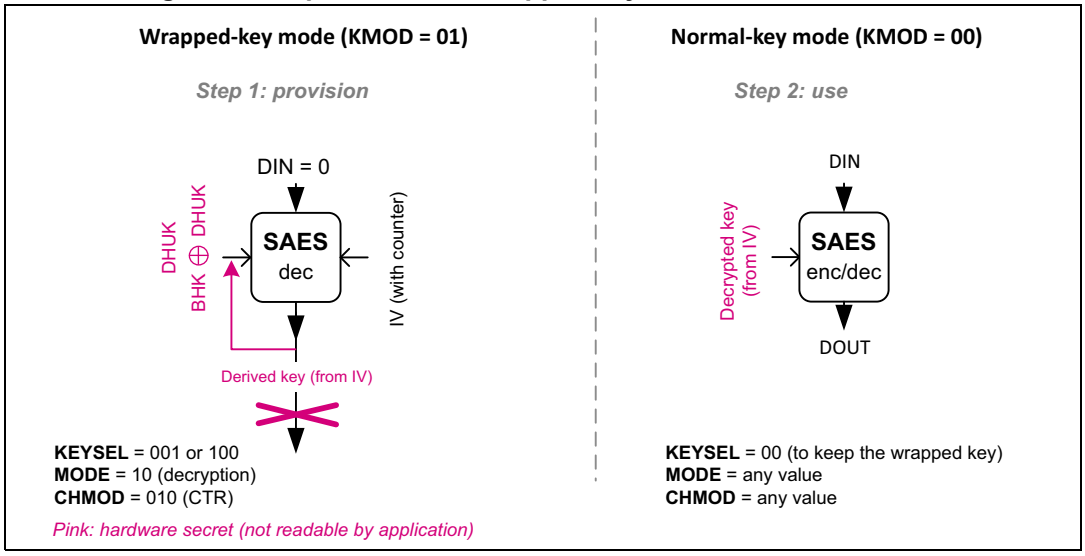
When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0], CHMOD[2:0] and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

Operation with wrapped keys for SAES in CTR mode

Figure 340 summarizes how to unwrap keys for SAES in CTR mode. To protect the derived key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by

writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 340. Operation with wrapped keys for SAES in CTR mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

The recommended sequence for SAES wrapped key mode using CTR is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the CTR chaining mode (CHMOD[2:0] at 0x2) in decryption mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. It is used for encryption key and for the key to share.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Write the initialization vector in SAES_IVRx registers, keeping the two least significant bits of SAES_IVR0 at zero.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 35.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_ISR register.
9. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The derived hardware secret key is available in SAES_KEYRx registers.
10. Repeat steps *8.* and *9.* if KEYSIZE is set.
11. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the hardware secret key derived from the public data in the SAES_IVRx registers is then immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Note: The configuration KMOD[1:0] at 0x1 (wrapped key), CHMOD[2:0] at 0x2 (CTR chaining), and MODE at 0x0 (encryption) disables the peripheral, by automatically clearing the EN bit of the SAES_*CR register.*

35.4.15 SAES operation with shared keys

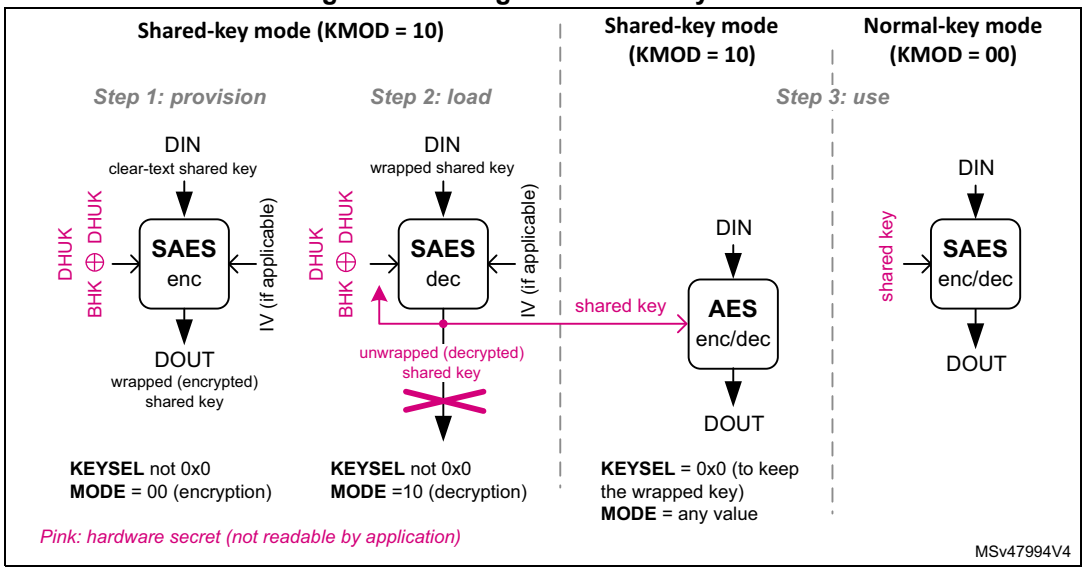
SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, the application software can make the AES keys available to the AES peripheral, without exposing them in clear-text (unencrypted).

Shared key sequences are too small to be suspended/resumed. SAES cannot unwrap a shared key using an unwrapped key.

Note: When a key stored in SAES is shared with AES, the protection given by KEYPROT bit is lost. The protection is detailed in Section 35.4.17: SAES key registers.

Figure 341 summarizes how to wrap or unwrap keys to share with *AES* peripheral. To protect the shared key, DHUK must be selected, by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 341. Usage of Shared-key mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KSHAREID, KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

In the step 3, AES represents the AES peripheral.

Key wrapping for AES peripheral

Before SAES can share a key with the AES peripheral, the key must be encrypted (wrapped) once. The encryption sequence of a shared key is the same as for a wrapped

key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 341*. See *Key wrapping for SAES* for details.

Note: Encryption in Shared-key mode is only supported when ECB or CBC is selected.

Key unwrapping for AES peripheral (shared key)

Each time SAES needs to share a key with the AES peripheral, shared encrypted key must be decrypted (unwrapped) in SAES, then loaded by AES. The overall sequence is described next.

Sequence in the SAES peripheral

The decryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 341*. See *Key unwrapping for SAES* for details.

In shared key mode when decryption mode is selected (MODE[1:0] at 0x2), a read access to the SAES_DOUTR register triggers a read error (RDERRF).

Note: Instead of being shared, a decrypted shared key can be used directly in SAES as the KEYSEL[2:0] bitfield is automatically cleared. In this case, KMOD[1:0] must be written with 0x0 (normal key mode).

Sequence in the AES peripheral

Once the shared key is decrypted in SAES key registers, it can be shared with the AES peripheral, while SAES peripheral remains in key sharing state, that is, with KMOD[1:0] at 0x2 and KEYVALID set. The sequence in the AES key share target peripheral is described in *AES key sharing with secure AES co-processor* of the corresponding section in this document. It can be run multiple times (for example, to manage a suspend/resume situation) as long as SAES is unused and duly remains in key sharing state.

Note: When KMOD[1:0] is at 0x2 and BUSY set in the AES peripheral, and KEYSIZE value of AES and SAES differs, the key sharing fails and the KEIF flag is raised in both peripherals.

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

35.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters the SAES peripheral with four successive 32-bit word writes into the SAES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of the SAES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 312* “no swapping” option for details.

Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the SAES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both SAES input and output.

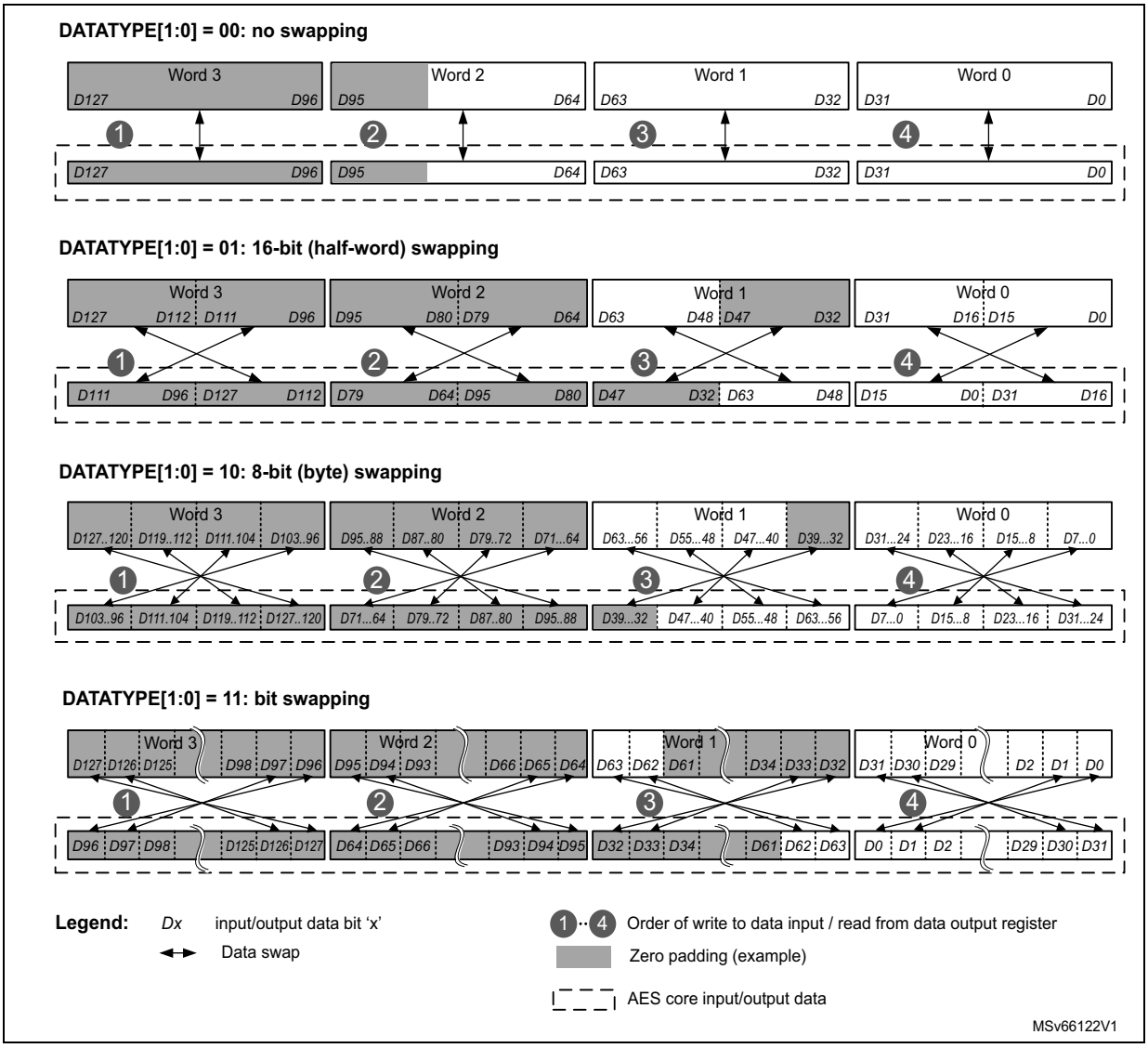
Note: The data in SAES key registers (SAES_KEYRx) and initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

The SAES data swapping feature is summarized in *Table 312* and *Figure 342*.

Table 312. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 342. 128-bit block construction according to the data type



Data padding

Figure 342 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

35.4.17 SAES key registers

The eight SAES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 313*. Reads are not allowed for security reason.

Note: In memory and in SAES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 313. Key endianness in SAES_KEYRx registers (128/256-bit keys)

SAES_KEYR7 [31:0]	SAES_KEYR6 [31:0]	SAES_KEYR5 [31:0]	SAES_KEYR4 [31:0]	SAES_KEYR3 [31:0]	SAES_KEYR2 [31:0]	SAES_KEYR1 [31:0]	SAES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
TAMP_BKP7R [31:0]	TAMP_BKP6R [31:0]	TAMP_BKP5R [31:0]	TAMP_BKP4R [31:0]	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYRx registers are ignored when SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), with KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- SAES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 35.4.19*).

Key selection

With KEYSEL[2:0] at 0x0, the application must write the key in the SAES_KEYRx registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a nonvolatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE information. Thanks to the key derivation function, the computed DHUK depends on the SAES usage (privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], CPU, and KEYSIZE context).

With KEYSEL[2:0] at 0x2, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 313*.

With KEYSEL[2:0] at 0x4, the XOR combination of DHUK and BHK is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR

registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 313*.

Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID is cleared. The recommended method to clear KEYVALID is to set IPRST. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] at 0x2).

For all KEYSEL[2:0] values, initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transfered to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see *Section 35.4.19*).

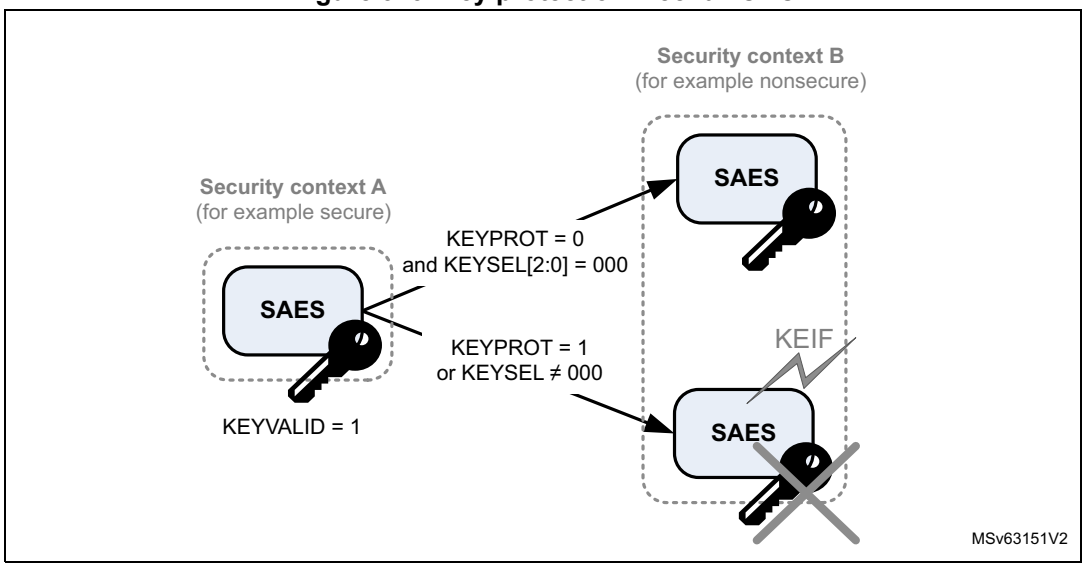
Note: DHUK, BHK and their XOR combination are not readable by any software (even secure).

Key protection

As depicted in *Figure 343*, when an application sets the KEYPROT bit before writing a key in SAES_KEYRx, any application executing in a different security context (that is, different security attribute) triggers a KEIF error flag upon access to any SAES register when KEYVALID is set.

Note: KEYSEL[2:0] values different from zero (normal key) automatically protect the key registers.

Figure 343. Key protection mechanisms



35.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in *Table 314*. They can only be written if the SAES peripheral is disabled (EN cleared).

Note: In memory and in SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 314. IVI bitfield spread over SAES_IVRx registers

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

35.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: Unless otherwise indicated, SAES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 35.5: SAES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: Unless otherwise indicated, SAES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 35.5: SAES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key protection error:** while KEYVALID is set, then if KEYPROT is set or KEYSEL[2:0] is different from zero, this error is triggered when an application executing in a security context different from the one used to load the key (that is, different security attribute) accesses SAES.

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 35.4.17: SAES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.
- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 35.4.15: SAES operation with shared keys* for details.
- **Hardware secret key loading error:** triggered upon failing load of DHUK or BHK into SAES. KEYSEL[2:0] at 0x1 (DHUK), 0x2 (BHK) or 0x4 (DHUK XOR BHK) is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to *Section 35.5: SAES interrupts*.

Upon a key selection error, clearing the KEIF flag automatically restarts the key selection process. Persisting problems (for example, RHUK load failing) may require a power-on reset.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring SAES.

RNG error interrupt flag (RNGEIF)

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the RNGEIE bit of the SAES_IER register is set. For more details, refer to *Section 35.5: SAES interrupts*.

Upon an RNG error:

- Make sure that the RNG peripheral is properly configured.
- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag then indicates the completion of the random number fetch from RNG.

Note: To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

Managing tamper errors

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any SAES co-processor processing.

To resume normal operation, reset the SAES peripheral through RCC or global reset.

35.5 SAES interrupts

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)
- RNG error (RNGEIF)

See *Section 35.4.19: SAES error management* for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

Table 315 gives a summary of the available features.

Table 315. SAES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
SAES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)
	RNG error flag	RNGEIF	RNGEIE	set RNGEIF(1)

1. Bit of the SAES_ICR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

35.6 SAES DMA requests

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with SAES can be found in *Appending data using DMA* subsection of *Section 35.4.5: SAES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 35.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

35.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 316. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles(1)
128-bit	Encryption or decryption(2)	ECB, CBC, CTR	**480**
	Key preparation	-	**145**
256-bit	Encryption or decryption(2)	ECB, CBC, CTR	**680**
	Key preparation	-	**230**

1. SAES kernel clock
2. Excluding key preparation time (ECB and CBC only).

Table 317. Processing latency for GCM and CCM (in SAES kernel clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	490	72(2)	480(3)	490
		CCM	490	490	800	490
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	650	72(2)	690(3)	650
		CCM	650	680	1350	650

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. SAES AHB clock cycles instead of kernel clock cycle (Galois multiplier only).
3. As a worst case in encryption mode, add extra 72 AHB clock cycles for the last block computation.

35.8 SAES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

35.8.1 SAES control register (SAES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: SAES peripheral software reset

Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:28 **KEYSEL[2:0]**: Key selection

The bitfield defines the source of the key information to use in the AES cryptographic core.

0x0: Software key, loaded in key registers SAES_KEYx

0x1: Derived hardware unique key (DHUK)

0x2: Boot hardware key (BHK)

0x4: XOR of DHUK and BHK

Others: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL[2:0] is different from zero, selected key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK when KEYVALID is cleared.

When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.

At the end of the decryption process, if KMOD[1:0] is other than zero, KEYSEL[2:0] is cleared.

With the bitfield value other than zero and KEYVALID set, the application cannot transfer the ownership of SAES with a loaded key to an application running in another security context (such as secure, nonsecure). More specifically, when security of an access to any register does not match the information recorded by SAES, the KEIF flag is set.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 27:26 **KSHAREID[1:0]**: Key share identification

This bitfield defines, at the end of a decryption process with KMOD[1:0] at 0x2 (shared key), which target can read the SAES key registers using a dedicated hardware bus.

0x0: AES peripheral

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the SAES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable and no specific use or protection applies to SAES_DINR and SAES_DOUTR registers.

0x1: Wrapped key for SAES mode. Key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

0x2: Shared key mode. After a successful decryption process (unwrapping), SAES key registers are shared with the peripheral described in KSHAREID[1:0] bitfield. This sharing is valid only while KMOD[1:0] at 0x2 and KEYVALID=1. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

Others: Reserved

With KMOD[1:0] other than zero, any attempt to configure the SAES peripheral for use by an application belonging to a different security context (such as secure or nonsecure) results in automatic key erasure and setting of the KEIF flag.

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 **KEYPROT**: Key protection

When set, hardware-based key protection is enabled.

0: When KEYVALID is set and KEYSEL[2:0] = 0 application can transfer the ownership of the SAES, with its loaded key, to an application running in another security context (such as nonsecure, secure).

1: When KEYVALID is set, key error flag (KEIF) is set when an access to any registers is detected, this access having a security context (for example, secure, nonsecure) that does not match the one of the application that loaded the key.

Attempts to write the bit are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the key length in bits of the key used by SAES.

0: 128-bit

1: 256-bit

When KMOD[1:0] is at 0x1 or 0x2, KEYSIZE also defines the length of the key to encrypt or decrypt.

Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **CPHASE[1:0]:** Chaining phase selection

This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode

This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode

This bitfield selects the SAES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the SAES_DINR register or read from the SAES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 35.4.16: SAES data registers and data swapping*.
0x0: No swapping (32-bit data).
0x1: Half-word swapping (16-bit data)
0x2: Byte swapping (8-bit data)
0x3: Bit-level swapping
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the SAES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the SAES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

The bit cannot be set as long as KEYVALID is cleared, or when SAES is in one of the following configurations:

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x3 (GCM)

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x2 (CTR), MODE[1:0] at 0x0 (encryption).

35.8.2 SAES status register (SAES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								r				r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in SAES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KEYSEL[2:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 35.4.17: SAES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether SAES is idle or busy.

0: Idle

1: Busy

SAES is flagged as idle when disabled (when EN is low) or when the last processing is completed.

SAES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), fetching random number from the RNG, or transferring a shared key to the target peripheral.

When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message. BUSY must also be cleared before selecting the GCM final phase.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected write to SAES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 0 Reserved, must be kept at reset value.

35.8.3 SAES data input register (SAES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the SAES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.

Reads return zero.

35.8.4 SAES data output register (SAES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

Read when KMOD[1:0] is at 0x1 or 0x2 while MODE[1:0] is at 0x2 and EN is set triggers a read error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the SAES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

35.8.5 SAES key register 0 (SAES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.

Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0 and KEYVALID is 0, writes to key registers are also ignored and they result in setting the KEIF bit of the SAES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

35.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

35.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

35.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

35.8.9 SAES initialization vector register 0 (SAES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in SAES_CR register

35.8.10 SAES initialization vector register 1 (SAES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

35.8.11 SAES initialization vector register 2 (SAES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

35.8.12 SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

35.8.13 SAES key register 4 (SAES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

35.8.14 SAES key register 5 (SAES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

35.8.15 SAES key register 6 (SAES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

35.8.16 SAES key register 7 (SAES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

35.8.17 SAES suspend registers (SAES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSP[31:0]**: Suspend data

SAES_SUSPRx registers contain the complete internal register states of the SAES when the CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 35.4.8: SAES suspend and resume operations* for more details.

Clearing EN bit of the SAES_CR register clears this register to zero.

SAES_SUSPRx registers are not used in other chaining modes than CCM.

35.8.18 SAES interrupt enable register (SAES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIE:** RNG error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

35.8.19 SAES interrupt status register (SAES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag

This read-only bit is set by hardware when an error is detected on RNG bus interface (for example bad entropy).

0: RNG bus is functional

1: Error detected on RNG bus interface (random seed fetching error)

The flag setting generates an interrupt if the RNGEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden.

0: No key error detected

1: Key information failed to load into key registers or the key register use is forbidden

The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set. It also clears the key registers and the KEYVALID flag in the SAES_SR register.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

KEIF is raised upon any of the following events:

–SAES fails to load the DHUK (KEYSEL[2:0] = 0x1 or 0x4).

–SAES fails to load the BHK (KEYSEL[2:0] = 0x2 or 0x4).

–AES fails to load the key shared by SAES peripheral (KMOD[1:0] = 0x2).

–KEYVALID is set and either KEYPROT is set or KEYSEL[2:0] is other than 0x0. The security context of the application that loads the key (secure or nonsecure) does not match the security attribute of the access to SAES_CR or SAES_DOUT. In this case, KEYVALID and EN bits are cleared.

–SAES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE set, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or reverse).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the SAES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the SAES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

35.8.20 SAES interrupt clear register (SAES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag clear

Application must set this bit to clear the RNGEIF status bit in SAES_ISR register.

Bit 2 **KEIF:** Key error interrupt flag clear

Setting this bit clears the KEIF status bit of the SAES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear

Setting this bit clears the RWEIF status bit of the SAES_ISR register, and clears both RDERRF and WRERRF flags in the SAES_SR register.

Bit 0 **CCF:** Computation complete flag clear

Setting this bit clears the CCF status bit of the SAES_ISR register.

35.8.21 SAES register map

Table 318. SAES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SAES_CR	IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]	Res.	CPHASE[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1]	CHMOD[0]	MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	SAES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	SAES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SAES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	SAES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	SAES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SAES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SAES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SAES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SAES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SAES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SAES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SAES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SAES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SAES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SAES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 318. SAES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SAES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SAES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SAES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SAES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SAES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	SAES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	SAES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	SAES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x300	SAES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
	Reset value																													0	0	0	0
0x304	SAES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x308	SAES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x309-0x3FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

36 Hash processor (HASH)

36.1 HASH introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512).

36.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO that can receive a single data block of size depending on the selected algorithm
- Automatic padding to complete the input bit string to fit digest minimum block size
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- 8 × 32-bit words (H0 to H15) for output message digest
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis
  - Reloadable digest registers

– Hashing computation suspend/resume mechanism, including DMA

36.3 HASH implementation

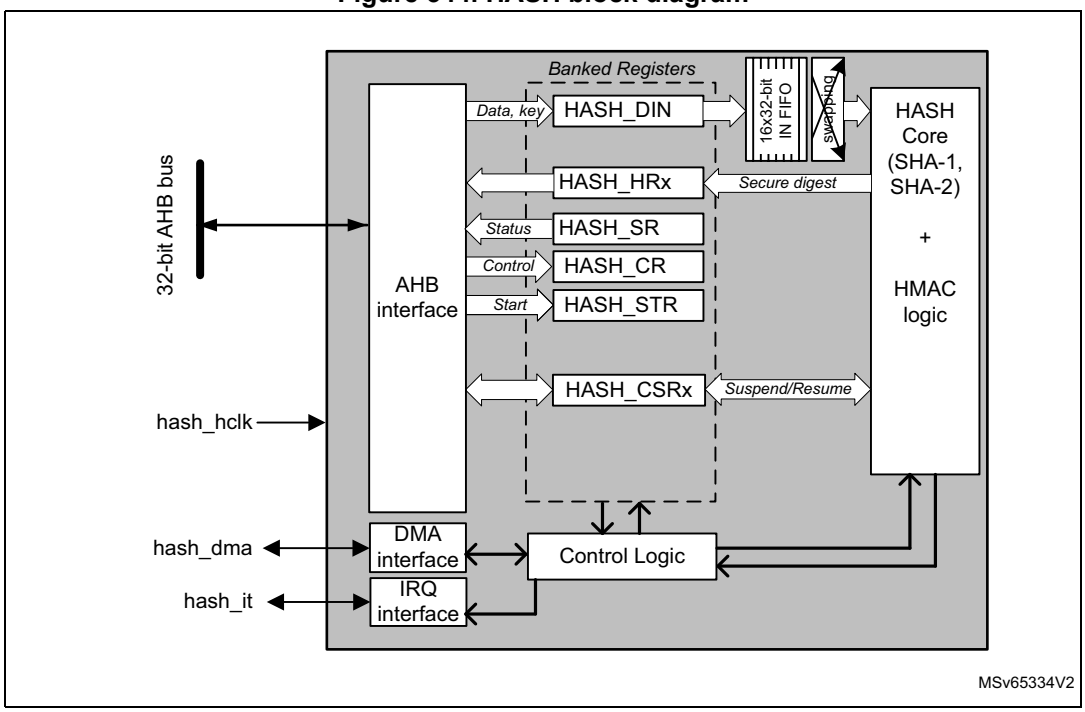
The devices have a single instance of HASH peripheral.

36.4 HASH functional description

36.4.1 HASH block diagram

Figure 344 shows the block diagram of the hash processor.

Figure 344. HASH block diagram



36.4.2 HASH internal signals

Table 319 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 319. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA burst request/ acknowledge

36.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see *Table 320*).

Table 320. Information on supported hash algorithms

Algorithm	Message digest size (in bits)	Block size (in bytes)(1)	Message length	Bit string message
SHA-1	160	64	$< 2^{64}$ bits	Yes
SHA2-224	224			
SHA2-256	256			
SHA2-384	384	128	$< 2^{128}$ bits	Yes
SHA2-512	512(2)			

1. Block size = (NBWE-1) * 4 bytes. NBWE[4:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017).

36.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “**01100001 01100010 01100011**” is represented by a 32-bit word **0x00636261**, and 8-bit words **0x61626300**.

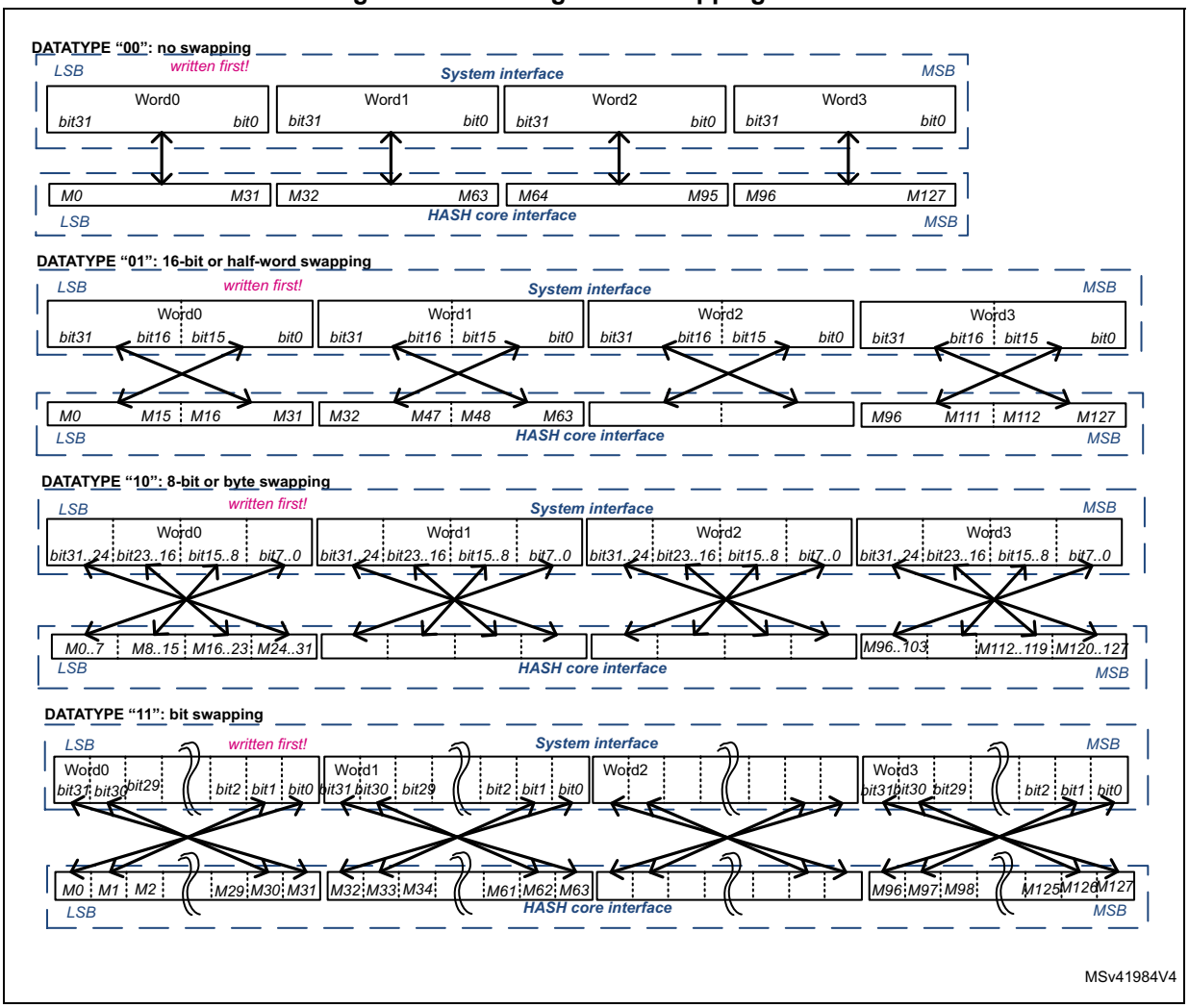
Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 345* shows how the hash processing core 32-bit data block M0...31 is

constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 345. Message data swapping feature



36.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in *Table 320: Information on supported hash algorithms*.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 36.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the

input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   - Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals **(NBWE-1) * 4**. This step is not required if the block size is already known (see *Table 320* for details).
   - Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   - When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to *Section 36.4.7: HMAC operation* for details.
   - Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - Data are filled by software:

     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.

     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 36.4.6* for details).
   - Data are filled as a single DMA transfer (MDMAT = 0):

     Partial digest computations are triggered automatically each time the FIFO is full.

     The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 36.4.6* for details).
   - Data are filled using multiple DMA transfers (MDMAT = 1):

     Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to

receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in *Table 321*. When set, the DMAEN bit is automatically cleared by hardware.

Table 321. Hash processor outputs

Algorithm	Valid output registers(1)	Most significant bit	Digest size (in bits)
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256
SHA2-384	HASH_H0 to HASH_H11	HASH_H0[31]	384
SHA2-512	HASH_H0 to HASH_H15		512(2)

1. Aliased HASH_HRAx registers are not used with SHA-3 or SHAKE algorithm (registers read as zero).
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)

For more information about HMAC detailed instructions, refer to *Section 36.4.7: HMAC operation*.

Aborting HASH operation

HASH can be disabled at any time by writing 0x4 to HASH_CR, after stopping any running DMA transfer to HASH (if applicable).

36.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed hashing sequences with padding are described in *Section 36.4.5: Message digest computing*. As mentioned in these sequences, the padding is done automatically using the information provided in NBLW[4:0].

Hashing example

Below is an example of SHA-1 digest computation with HASH. Message processing and padding are defined in Federal Information Processing Standards PUB 180-4.

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24 (‘**U**’ means don’t care):

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “**1**” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are automatically appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, **L**, in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where ‘**U**’ means don’t care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x100** is written to the HASH_STR register to run the final digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D
```

36.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message)= Hash((Key | pad) XOR opad |
               Hash((Key | pad) XOR ipad | message))
```

Where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the byte size of the underlying hash function data block (n = 64 when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n*, the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.
3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 36.4.5: Message digest computing*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 36.4.11: HASH processing time.

Endianness management details can be found in Section 36.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "**`Sample message for keylen = blocklen`**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

   ```
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F
   ```

2. **Message** input (length = 34, that is, padding required). HASH_STR must be set to **`0x20`** to start message padding and inner hash computation (see **'U**' as don't care)

   ```
   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
   6E3D626C 6F636B6C 656EUUUU
   ```

3. **Outer hash key** input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.
4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

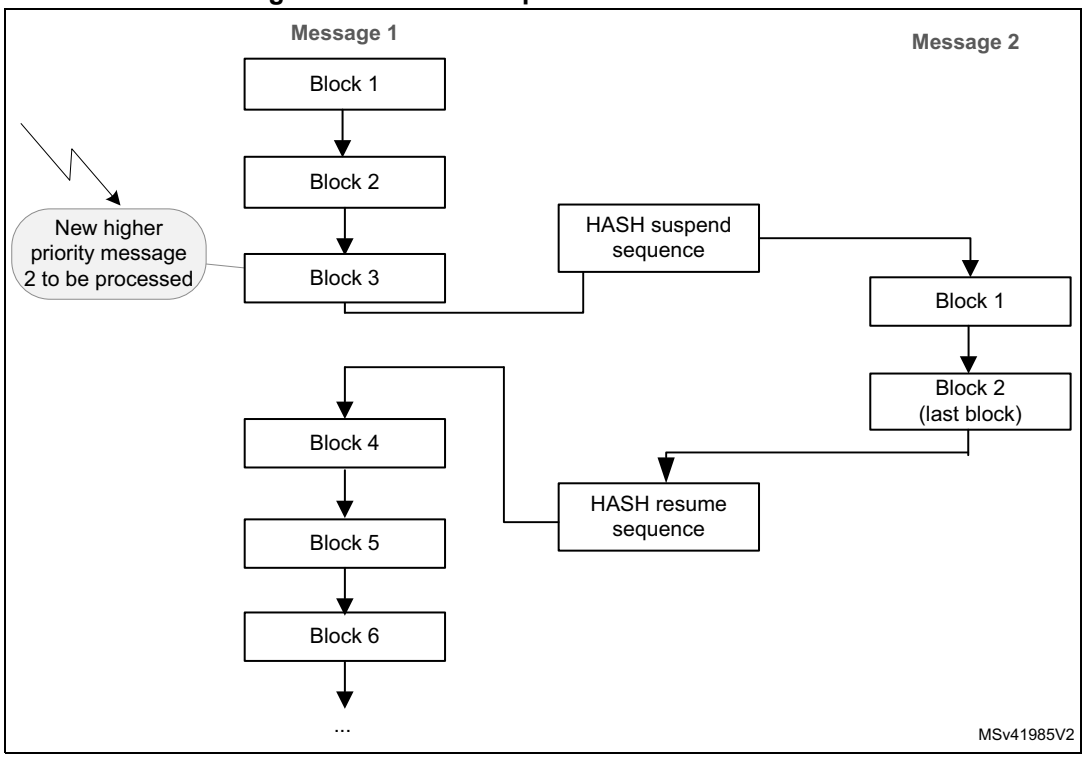
   ```
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
   ```

36.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 346*.

Figure 346. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 346*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

36.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to *Section 36.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW = Len(Message)% 32` where "**`x%32`**" gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

36.4.10 HASH error management

No error flags are generated by the hash processor.

36.4.11 HASH processing time

Table 322 summarizes the time required to process an intermediate block for each mode of operation.

Table 322. Processing time (in clock cycle)

Mode of operation	Block size (in bytes)	FIFO load(1)	Computation phase	Total
SHA-1	64	16	66	**82**
SHA2-224		16	50	**66**
SHA2-256				
SHA2-384	128	32	66	**98**
SHA2-512(2)				

1. Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

36.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 323* gives a summary of the available features.

Table 323. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

36.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

36.6.1 HASH control register (HASH_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:0]				LKEY
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	w		

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **ALGO[3:0]**: Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

0000: SHA-1

0001: reserved

0010: SHA2-224

0011: SHA2-256

1100: SHA2-384

1101: SHA2-512/224

1110: SHA2-512/256

1111: SHA2-512

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see *Table 320: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.

1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield.

This bitfield is read-only.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **MODE:** Mode selection

This bit selects the normal or the keyed HMAC mode for the selected algorithm. This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

0: Hash mode selected

1: HMAC mode selected. LKEY bit must be set if the key being used is longer than the algorithm block size.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of the data entered into the HASH_DIN register:
00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

Setting this bit enables DMA transfers. After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.
Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.
Setting INIT bit does not clear DMAE bit.
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

Bit 2 **INIT:** Initialize message digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit resets the hash processor core, so that HASH is ready to compute the message digest of a new message. Clearing this bit has no effect.

Bits 1:0 Reserved, must be kept at reset value.

36.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see *Table 320: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.

Do not write to this register when BUSY is set.

Reading this register returns zeros.

36.6.3 HASH start register (HASH_STR)

Address offset: 0x008

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							w				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.

Do not set DCAL when BUSY is set.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping.

The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

Note:

36.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512 digest result

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

Note: When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 36.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 36.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 5 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 36.6.4: HASH digest registers* introduction.

36.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

36.6.6 HASH status register (HASH_SR)

Address offset: 0x024

Reset value: 0x0011 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	Res.	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r		r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]:** Number of words expected
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.
NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 **DINNE:** DIN not empty
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.
0: No data are present in the data input buffer
1: The input buffer contains at least one word of data

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)

1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

36.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 102)

Reset value: 0x0022 0002 (HASH_CSR0)

Reset value: 0x0000 0000 (others)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSx[31:0]:** Context swap x

Refer to *Section 36.6.7: HASH context swap registers* introduction.

36.6.8 HASH register map

Table 324. HASH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:2]		ALGO[1:0]		LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	Reset value												0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0		
0x004	HASH_DIN	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x00C + 0x4 * x, (x = 0 to 4) Last address: 0x01C	HASH_HRAx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x024	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NBWE[4:0]					DINNE		NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value												1	0	0	0	1	0		0	0	0	0	0						0	0	0	1
Reserved																																	
0x0F8	HASH_CSR0	CSR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 324. HASH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0FC	HASH_CSR1	CSR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HASH_CSR2	CSR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x290	HASH_CSR102	CSR102[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310 + 0x4 * x, (x = 0 to 15) Last address: 0x34C	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

37 Public key accelerator (PKA)

This section only applies to the STM32G54x (fully) and STM32G53x (only the ECDSA signature verification part) devices.

37.1 PKA introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, the accelerator performs all computations with no further hardware or software elaboration required to process the inputs or the outputs.

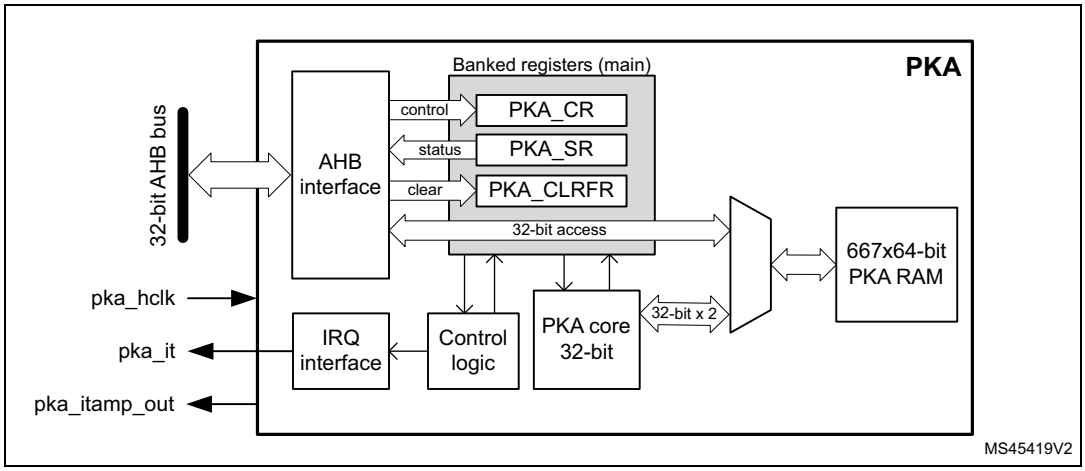
37.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)

37.3 PKA functional description

37.3.1 PKA block diagram

Figure 347. PKA block diagram



Note: Ignore pka_itamp_out, it is not applicable.

37.3.2 PKA internal signals

Table 325 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 325. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request

37.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 667 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to *Section 37.3.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

37.3.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information can be found in *Section 37.5.1*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 326* and *Table 327*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 326. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R^2 mod *n*	*Section 37.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 37.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 37.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 37.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 37.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x08	001000	Modular inversion A^{-1} mod n	*Section 37.4.7*
0x0D	001101	Modular reduction A mod n	*Section 37.4.8*
0x09	001001	Arithmetic addition A+B	*Section 37.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 37.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 37.4.11*
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 37.4.12*
0x07	000111	RSA CRT exponentiation	*Section 37.4.13*

Table 327. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 37.4.14*
0x20	100000	ECC scalar multiplication kP	*Section 37.4.15*
0x23	100011	ECC complete addition	*Section 37.4.18*
0x24	100100	ECDSA sign	*Section 37.4.16*
0x26	100110	ECDSA verification	*Section 37.4.17*
0x27	100111	ECC double base ladder	*Section 37.4.19*
0x2F	101111	ECC projective to affine	*Section 37.4.20*

Each of these operating modes has an associated code that has to be written to the MODE[5:0] bitfield in the PKA_CR register. If the application selects any value that is not documented below the write to MODE[5:0] bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Montgomery space and fast mode operations

For efficiency reasons, the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming, the application can decide to use a faster mode of operation during which the precomputed Montgomery parameter is supplied before starting the operation. The performance improvement is detailed in *Section 37.5.2: Computation times*.

37.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions such as:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

The following publications specify the listed functions:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

Refer to these documents for full details. This document limits itself to describing principles.

RSA key pair

For the following RSA operations, a public key and a private key information are defined as follows:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively, this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information, refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (n, e), must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer m, with $0 \leq m < n$ and (m, n) being coprimes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key d, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext m is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
3. Convert the integer message representative m to an encoded message EM.
4. Recover message M = DECODE(EM), where DECODE is a decoding method.

To accelerate these operations with the PKA, use *Modular exponentiation* $A^e \bmod n$ if the private key is d, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

Elliptic curve selection

For the following ECC operations, the curve parameters are defined as follows:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 37.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, $n \times G$ = identity element O).

ECDSA message signature generation

This section describes the ECDSA (elliptic curve digital signature algorithm) signature generation function principle.

Alice, to sign a message m using her private key integer d_A, goes through the following steps:

1. Calculate e = HASH(m), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order n.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.

7. The signature is the pair (r, s).

The PKA accelerates the steps 4. to 7. by using:

- *ECDSA sign*, or
- All of the following operations:
  - *ECC Fp scalar multiplication* $k \times P$
  - *Modular reduction* A mod n
  - *Modular inversion* A^{-1} mod n
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

This section describes the ECDSA (elliptic curve digital signature algorithm) signature verification function principle.

Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A. Bob can verify that Q_A is a valid curve point by going through the following steps:

1. Check that Q_A is not equal to the identity element O
2. Check that Q_A is on the agreed curve
3. Check that $n \times Q_A = O$.

Then Bob executes the following procedure:

1. Verify that r and s are integer in [1, n-1]
2. Calculate e = HASH(m), where HASH is the agreed cryptographic hash function
3. Let z be the L_n leftmost bits of e
4. Calculate $w = s^{-1}$ mod n
5. Calculate $u_1 = zw$ mod n and $u_2 = rw$ mod n
6. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. The signature is valid if $r = x_1$ (mod n), it is invalid otherwise.

The PKA accelerates the steps 4. to 7. by using *ECDSA verification*.

37.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit in the PKA_CR register enables the PKA peripheral. The PKA becomes available when the INITOK bit is set in the PKA_SR register. When EN = 0, the PKA peripheral is kept under reset, with its memory still accessible by the application through the AHB interface.

Note: When the PKA is in the process of clearing its memory, the EN bit cannot be set.

When setting the EN bit in PKA_CR, make sure that the MODE[5:0] bitfield value corresponds to an authorized PKA operation (see OPERRF in Section 37.3.7).

Clearing the EN bit while a calculation is in progress aborts the operation. In this case, the content of the PKA memory is not guaranteed.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 37.4*.

Executing a PKA operation

To execute any of the supported PKA operations, proceed as follows:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE[5:0] bitfield of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1), any access by the application to PKA RAM is ignored, and the RAMERRF flag is set in PKA_SR.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error and prevents the PROCENDF flag from being set and the procedure step *3* from being completed. See *Section 37.3.7* for details.

Using precomputed Montgomery parameters (PKA fast mode)

As explained in *Section 37.3.4*, when computing many operations with the same modulus, it can be beneficial for the application to compute the corresponding Montgomery parameter only once (see, for example, *Section 37.4.5*). This is referred to as *fast mode*.

To use the PKA peripheral in fast mode, follow this recommended procedure:

1. Load the modulus size and value information into the PKA RAM. Refer to *Section 37.5.1* for detail.
2. Configure the PKA_CR register to *Montgomery parameter computation* mode (MODE[5:0] = 0x1), then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set. Then, read the corresponding Montgomery parameter from the PKA memory and clear the PROCENDF bit by setting PROCENDFC in PKA_CLRFR.
4. Continue with the required PKA operation, loading the Montgomery information R2 mod *m* on top of the regular input data. Relevant addresses are provided in *Section 37.4*.

37.3.7 PKA error management

The following error flags in the PKA_SR register are set to indicate errors that may occur when using the PKA:

- **ADDRERRF** flags that the access to PKA RAM falls outside the expected range.
- **RAMERRF** flags an AHB access to the PKA RAM occurring while the PKA core uses the PKA RAM. While the flag is set, reads form the PKA RAM return 0 and writes are ignored.
- **OPERRF** flags that the selected operating mode using MODE[5:0] bitfield is not listed in PKA operating modes (see the bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). While the flag is set, writes to the MODE[5:0] bitfield are ignored.

For each error flag, the PKA generates an interrupt if enabled with the corresponding bit in the PKA_CR register (see *Section 37.6* for details).

The ADDRERRF, OPERRF, and RAMERRF error flags are cleared by setting the ADDRERRFC, OPERRFC, and RAMERRFC bit in PKA_CLRFR, respectively. The OPERRF error flag must be cleared before a new operation is initiated via the PKA_CR register.

Note: The PKA can be re-initialized at any moment by clearing the EN bit in the PKA_CR register.

37.4 PKA operating modes

37.4.1 Introduction

The operations supported by the PKA are detailed in the following subsections, which define the format of both input data and results stored in the PKA RAM.

Warning: Before starting any operation, ensure that all input parameters to the PKA are valid and consistent, as the PKA assumes their correctness. Additionally, input parameters must not exceed the operand sizes specified in the operation tables.

The following information applies to all PKA operations:

- The PKA core processes 64-bit words in its RAM. Accordingly, in the PKA section of this document, 'word' denotes a 64-bit unit.
- When writing an element as input in the PKA RAM, append an additional 64-bit word of zeros after the most significant word, except when the operand size is fixed at one word.
- All RAM storage addresses refer to the least significant word of the data. To obtain the actual address, the application must add the base address of the PKA to the indicated offset.
- Supported operand sizes are:
  - **ROS (RSA operand size)**: Data size is calculated as *(rsa_size / 64) + 1* words, where *rsa_size* is the modulus length in bits. For example, a 1024-bit RSA operand corresponds to 17 words (1088 bits).
  - **EOS (ECC operand size)**: Data size is *(ecc_size / 64) + 1* words, where *ecc_size* is the prime modulus length in bits. For example, a 192-bit ECC operand corresponds to four words (256 bits).

  Note: Both ROS and EOS include the additional zero word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Any fractional results in the formulas must be rounded to the nearest integer, as the PKA core processes data in 64-bit words.

Note: The maximum ROS is 66 words (4160-bit maximum exponent size), while the maximum EOS is 11 words (640-bit max operand size).

Example 1: Understanding endianness in PKA memory during ECC Fp scalar multiplication preparation

Write the *x*-coordinate of point P for an ECC P256 curve (EOS = 5 words) by placing the least significant bit at bit 0 of the word at address offset 0x578, and the most significant bit at bit 63 of the word at address offset 0x590. Then, write a zero word (0x0000000000000000) at address offset 0x598.

Example 2: Writing the curve coefficient '*a*' for ECC Fp scalar multiplication

To write the value *a* = -3 for a curve with a modulus length of 224 bits (rounded up to five 64-bit words), write the PKA memory with data as in the following table:

Table 328. Example of *'a'* curve coefficient for ECC Fp scalar

RAM offset	Value	Note
0x410	0x0000000000000001	Curve coefficient *a* sign without extra word
0x418	0x0000000000000011	Value of |*a*| LSB
0x420	0x0000000000000000	-
0x428	0x0000000000000000	-
0x430	0x0000000000000000	Value of |*a*| MSB
0x438	0x0000000000000000	Additional zero word

37.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod n) used by the PKA to convert operands into the Montgomery residue system representation. This operation can be very useful for fast mode operation because passing the Montgomery parameter as input saves the time for its computation.

Note: This operation also supports ECC curves when specifying the prime modulus length and EOS size.

The following table summarizes operation instructions for Montgomery parameter computation.

Table 329. Montgomery parameter computation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	ROS
OUT	Result: R^2 mod *n*	-	RAM@0x620	

37.4.3 Modular addition

The following table summarizes operation instructions for modular addition operation that calculates *A* + *B* mod *n*.

Table 330. Modular addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A + B \bmod n$	($0 \le$ result $< n$)	RAM@0xE78	

37.4.4 Modular subtraction

Modular subtraction operation calculates:

- ***A* - *B* mod *n***, when $A \ge B$
- ***A* + *n* - *B* mod *n***, when $A < B$

The following table summarizes the operation instructions.

Table 331. Modular subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A - B \bmod n$	($0 \le$ result $< n$)	RAM@0xE78	

37.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications, the PKA accelerates a multiplication that has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication $A \times B \bmod n$

The method to perform these operations is described next. The *x* function is this operation, and the *A*, *B*, and *C* operands are in the natural domain.

1. Inward or outward conversion to or from the Montgomery domain
   a) Assuming that *A* is an integer in the natural domain:
      – Compute *r2modn* using *Montgomery parameter computation*.
      – The result of *AR* = *A* × *r2modn* mod *n* is *A* in the Montgomery domain.
   b) Assuming that *BR* is an integer in the Montgomery domain:
      – The result of *B* = *BR* × 1 mod *n* is *B* in the natural domain.
      – Similarly, the *AR* value computed in *a)* can be converted into the natural domain by computing *A* = *AR* × 1 mod *n.*
2. Simple modular multiplication *A* × *B* mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute *AR* = *A* × *r2modn* mod *n*. The output is in the Montgomery domain.
   c) Compute *AB* = *AR* × *B* mod *n*. The output is in the natural domain.
3. Multiple modular multiplication *A* × *B* × *C* mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute *AR* = *A* × *r2modn* mod *n*. The output is in the Montgomery domain.
   c) Compute *BR* = *B* × *r2modn* mod *n*. The output is in the Montgomery domain.
   d) Compute *ABR* = *AR* × *BR* mod *n*. The output is in the Montgomery domain.
   e) Compute *CR* = *C* × *r2modn* mod *n*. The output is in the Montgomery domain.
   f) Compute *ABCR* = *ABR* × *CR* mod *n*. The output is in the Montgomery domain.
   g) (optional) Repeat the two previous steps if multiplying with additional operands is required.
   h) Compute *ABC* = *ABCR* × 1 mod *n* to retrieve the result in the natural domain.

The following table summarizes the operation instructions.

Table 332. Montgomery multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	(0 ≤ *A* < *n*)	RAM@0xA50	ROS
	Operand *B*	(0 ≤ *B* < *n*)	RAM@0xC68	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: *A* × *B* mod *n*(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

37.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It calculates of A^e mod *n*.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

Operation instructions for modular exponentiation are summarized in *Table 333* (normal mode) and *Table 334* (fast mode). Fast mode usage is explained in *Section 37.3.6*.

Table 333. Modular exponentiation (normal mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A^e \bmod n$	($0 \le$ result $< n$)	RAM@0x838	

Table 334. Modular exponentiation (fast mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
IN/OUT	Montgomery parameter $R^2 \bmod n$	(mandatory)	RAM@0x620	
OUT	Result: $A^e \bmod n$	($0 \le$ result $< n$)	RAM@0x838	

37.4.7 Modular inversion

The modular inversion operation calculates multiplicative inverse $A^{-1} \bmod n$. If the modulus *n* is prime, for all values of *A* ($1 \le A < n$) modular inversion output is valid. If the modulus *n* is not prime, *A* has an inverse only if the greatest common divisor between *A* and *n* is 1.

If the operand *A* is a divisor of the modulus *n* the result is a multiple of a factor of *n*.

The following table summarizes the operation instructions.

Table 335. Modular inversion

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result: $A^{-1} \bmod n$	$0 < \text{result} < n$	RAM@0xE78	

37.4.8 Modular reduction

The modular reduction operation calculates the remainder of *A* divided by *n*.

The following table summarizes the operation instructions.

Table 336. Modular reduction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand *A*	($0 \le A < 2n < 2^{4160}$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result *A* mod *n*	($0 < \text{result} < n$)	RAM@0xE78	

37.4.9 Arithmetic addition

The arithmetic addition operation calculates *A* + *B*.

The following table summarizes the operation instructions.

Table 337. Arithmetic addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x09	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* + *B*	($0 \le \text{result} < 2^{M+1}$)	RAM@0xE78	ROS + 1

37.4.10 Arithmetic subtraction

The arithmetic subtraction operation calculates:

- ***A - B****,* when $A \geq B$
- $\boldsymbol{A + 2^{int(M / 32) \times 32 + 1} - B}$*,* when $A < B$ and the residue of $M / 32$ is greater than zero
- $\boldsymbol{A + 2^{int(M / 32) \times 32} - B}$*,* when $A < B$ and the residue of $M / 32$ is zero

For the last two list items, the 32-bit word following the most significant word of the output is 0xFFFF FFFF, as the result is negative.

The following table summarizes the operation instructions.

Table 338. Arithmetic subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0A	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \leq < A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \leq < B < 2^M$)	RAM@0xC68	
OUT	Result: *A* - *B*	($0 \leq$ result $< 2^M$)	RAM@0xE78	

37.4.11 Arithmetic multiplication

The arithmetic multiplication operation calculates A × B.

The following table summarizes the operation instructions.

Table 339. Arithmetic multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0B	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \leq A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \leq B < 2^M$)	RAM@0xC68	
OUT	Result: *A* × *B*	($0 \leq$ result $< 2^{2M}$)	RAM@0xE78	2xROS

37.4.12 Arithmetic comparison

The arithmetic comparison operation yields the following results:

- **0xED2C**, when $A = B$
- **0x7AF8**, when $A > B$
- **0x916A**, when $A < B$

The following table summarizes the operation instructions.

Table 340. Arithmetic comparison

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0C	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	If result *A* = *B*	0xED2C, 0x7AF8, or 0x916A	RAM@0xE78	64 bits

37.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- p and q are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation m = $A^d \pmod{pq}$ more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod q$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq \bmod pq$

Operation instructions for computing CRT exponentiation $A^d \bmod pq$ are summarized in *Table 341*.

Table 341. CRT exponentiation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	($0 < d_P < 2^{M/2}$)	RAM@0x730	ROS / 2
	Operand d_Q	($0 < d_Q < 2^{M/2}$)	RAM@0xE78	
	Operand q_{inv}	($0 < q_{inv} < 2^{M/2}$)	RAM@0x948	
	Prime $p^{(1)}$	($0 < p < 2^{M/2}$)	RAM@0xB60	
	Prime $q^{(1)}$	($0 < q < 2^{M/2}$)	RAM@0x1088	
IN	Operand *A*	($0 < A < 2^M$)	RAM@0x12A0	ROS
OUT	Result: $A^d \bmod pq$	($0 \le$ result $<$ pq)	RAM@0x838	

1. Must be different from 2.

37.4.14 Point on elliptic curve Fp check

This operation checks whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where a and b are elements of the curve.

The following table summarizes the operation instructions.

Table 342. Point on elliptic curve *Fp* check

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x28	PKA_CR	6 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, $\|a\| < p$)	RAM@0x418	EOS
	Curve coefficient *b*	($\|b\| < p$)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2640$)	RAM@0x470	
	Point P coordinate *x*	($x < p$)	RAM@0x578	
	Point P coordinate *y*	($y < p$)	RAM@0x5D0	
	Montgomery parameter $R^2 \bmod n$	-	RAM@0x4C8	
OUT	Result: point *P* on curve	0xD60D: point on curve 0xA3B7: point not on curve 0xF946: *x* or *y* coordinate is not smaller than modulus *p*	RAM@0x680	64 bits

37.4.15 ECC Fp scalar multiplication

This operation calculates $k \times P$ (x_P, y_P), where P is a point on a curve over prime fields and x is the elliptic curve scalar point multiplication. The result is a point that belongs to the same curve or a point at infinity.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

The following table summarizes the operation instructions.

Table 343. ECC Fp scalar multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x20	PKA_CR	6 bits
IN	Scalar multiplier *k* length	(in bits, not null, 8 < value < 640)	RAM@0x400	64 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	

Table 343. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (note)	Storage	Size
IN	Curve coefficient $\|a\|$	(absolute value, $\|a\| < p$)	RAM@0x418	EOS
	Curve modulus value p	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Scalar multiplier k	($0 \le k < n$)	RAM@0x520	
	Point P coordinate x_P	($x_P < p$)	RAM@0x578	
	Point P coordinate y_P	($y_P < p$)	RAM@0x5D0	
OUT	Result: $k \times P$ coordinate x'	(result < p)	RAM@0x578	EOS
	Result: $k \times P$ coordinate y'	(result < p)	RAM@0x5D0	

When performing this operation the following special cases must be noted:

- For $k = 0$ this function returns a point at infinity (0, 0) if curve parameter b is nonzero, (0, 1) otherwise. For k different from 0 it might happen that a point at infinity is returned. When the application detects this behavior, a new computation must be carried out.
- For $k < 0$ (that is, a negative scalar multiplication is required) the multiplier absolute value
  $k = |-k|$ must be provided to the PKA. After the computation completion, the formula $-P = (x, -y)$ can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

37.4.16 ECDSA sign

Table 344 (input parameters) and *Table 345* (output parameters) summarize the ECDSA signing operation.

Upon an output error different from 0xD60D, the application must generate a new k and repeat the ECDSA sign operation.

Note: As long as this operation is ongoing (BUSY = 1), the PKA control register and the PKA memory are not accessible. The access is restored when the PKA clears the BUSY bit.

Table 344. ECDSA sign - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x24	PKA_CR	6 bits
	Curve prime order *n* length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer *k*(1)	(0 ≤ *k* < n)	RAM@0x520	
	Curve base point *G* coordinate *x*	(*x* < *p*)	RAM@0x578	
	Curve base point *G* coordinate *y*	(*y* < *p*)	RAM@0x5D07	
	Hash of message *z*	(hash size equal to *nlen*)(2)	RAM@0xED0	
	Private key *d*	(0 < *d*)	RAM@0xF28	
	Curve prime order *n*	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases *k* can be deterministically generated.
2. To match the hash parameter size to the length of the curve prime order *n,* pad the result with zeros or use the hash truncation.

Table 345. ECDSA sign - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Signature part *r*	(0 < *r* < *n*)	RAM@0x730	EOS
	Signature part *s*	(0 < *s* < *n*)	RAM@0x788	
ERROR	Result of signature	0xD60D: successful computation, no error 0xCBC9: failed computation 0xA3B7: signature part r is equal to 0 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note:

Extended ECDSA support

The PKA also supports the extended ECDSA signature, which accepts the same inputs as the standard ECDSA signature (see *Table 344*) and produces similar outputs (see *Table 345*), with the addition of the point *kG* coordinates. The following table defines these additional outputs.

Table 346. Extended ECDSA sign - additional outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Curve point kG coordinate x_1	($0 \le x_1 < p$)	RAM@0x1400	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

37.4.17 ECDSA verification

Table 347 (input parameters) and *Table 348* (output parameters) summarize the ECDSA verification operation.

Upon an output error different from 0xD60D, the signature is not verified.

Table 347. ECDSA verification - inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x26	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus p length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient $\|a\|$	(absolute value, $\|a\| < p$)	RAM@0x470	EOS
	Curve modulus value p	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x4D0	
	Curve base point G coordinate x	($x < p$)	RAM@0x678	
	Curve base point G coordinate y	($y < p$)	RAM@0x6D0	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part r	($0 < r < n$)	RAM@0x10E0	
	Signature part s	($0 < s < n$)	RAM@0xC68	
	Hash of message z	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order n	(integer prime)	RAM@0x1088	

1. To match the hash parameter size to the length of the curve prime order n, pad the result with zeros or use the hash truncation.

Table 348. ECDSA verification - outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– ($0 < r < n$)	RAM@0x578	EOS

37.4.18 ECC complete addition

The ECC complete addition operation calculates the sum of two points on an elliptic curve.

Operation instructions are summarized in *Table 349*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. To convert resulting point to Jacobian coordinates conversion (x, y)= (X / Z^2, Y / Z^3) can be used.

Table 349. ECC complete addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient $\|a\|$	(absolute value, $\|a\| < p$)	RAM@0x418	
	First point P coordinate *X*	($x < p$)	RAM@0x628	
	First point P coordinate *Y*	($y < p$)	RAM@0x680	
	First point P coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point Q coordinate *X*	($x < p$)	RAM@0x730	
	Second point Q coordinate *Y*	($y < p$)	RAM@0x788	
	Second point Q coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *X*	($x < p$)	RAM@0xD60	
	Result coordinate *Y*	($y < p$)	RAM@0xDB8	
	Result coordinate *Z*	($z < p$)	RAM@0xE10	

37.4.19 ECC double base ladder

ECC double base ladder operation consists in the computation of $k \times P + m \times Q$, where (P, Q) are two points on an elliptic curve and (k, m) are two scalars. Operation instructions are summarized in *Table 350*.

If the resulting point is the point at infinity (error code 0xA3B7), the resulting coordinate equals to (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 350. ECC double base ladder

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer *k*	($0 < k < 2^{640}$)	RAM@0x520	
	Integer *m*	($0 < m < 2^{640}$)	RAM@0x578	
	First point *P* coordinate *X*	($x < p$)	RAM@0x628	
	First point *P* coordinate *Y*	($y < p$)	RAM@0x680	
	First point *P* coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point *Q* coordinate *X*	($x < p$)	RAM@0x730	
	Second point *Q* coordinate *Y*	($y < p$)	RAM@0x788	
	Second point *Q* coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *x*	($x < p$)	RAM@0x578	EOS
	Result coordinate *y*	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

37.4.20 ECC projective to affine

ECC projective to affine operation calculates the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (*X*, *Y*, *Z*), it calculates the affine coordinates (*x*, *y*) = (*X* / *Z*, *Y* / *Z*).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 351*.

Table 351. ECC projective to affine

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate *X* (projective)	($x < p$)	RAM@0xD60	
	Point P coordinate *Y* (projective)	($y < p$)	RAM@0xDB8	
	Point P coordinate *Z* (projective)	($z < p$)	RAM@0xE10	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Point P coordinate *x* (affine)	($x < p$)	RAM@0x578	
	Point P coordinate *y* (affine)	($y < p$)	RAM@0x5D0	
ERROR	Error code	Point not at infinity: 0xD60D Point at infinity: 0xA3B7	RAM@0x680	64 bits

37.5 Example of configurations and processing times

37.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b$ (mod p).

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field *GF(p)*
- the (usually prime) integer *n*, the order of the group generated by *G*, defined below
- the base point of the curve *G*, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. ***a* defined as *p*-|*a*|** in the finite field *GF(p)*, for example ***p*-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 352 summarizes the family of curves supported by PKA for ECC operations.

Table 352. Family of supported curves for ECC operations

Curve name	Standard	Reference	
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4	
P-224			
P-256			
P-384			
P-521			
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

37.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 353. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	124600	491000	684000	1133200
	Fast	22700	82000	178000	311000
17	Normal	135700	531400	772400	1288000
	Fast	33800	122500	266500	465800
$2^{16} + 1$	Normal	180000	693700	1126200	1907200
	Fast	78200	284700	620400	1085000
1024	Normal	5850000	-	-	-
	Fast	5748000	-	-	-
	CRT(1)	1775000	-	-	-
2048	Normal	-	42240000	-	-
	Fast	-	41832000	-	-
	CRT(1)	-	11670000	-	-
3072	Normal	-	-	136830000	-
	Fast	-	-	136325000	-
	CRT(1)	-	-	36886000	-
4096	Normal	-	-	-	316000000
	Fast	-	-	-	315226000
	CRT(1)	-	-	-	84577000

1. CRT stands for chinese remainder theorem optimization (MODE[5:0] bitfield= 0x07).

Table 354. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
830000	1248000	2480000	4257000	6849000	14529000	16565000	25795000

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 355. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
885000	1329000	2624000	4536000	7232000	15334000	17669000	26978000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 356. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
1011000	1495000	2938000	5014000	7979000	16804000	19254000	29582000

Table 357. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
967000	1419000	2768000	4784000	7547000	15854000	18257000	28257000

Table 358. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
47600	78000	148300	253000	419000	838400	1049300

Table 359. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
10000	12000	18000	26000	39000	63000	89000

Table 360. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
3400	4200	6100	8300	10900	17200	-	-

Table 361. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
8600	8710	11870	17000	102000	410000	506000	822000

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

37.6 PKA interrupts

There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 37.3.7*
2. Access to unmapped address (ADDRERRF), see *Section 37.3.7*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 37.3.7*
4. PKA end of operation (PROCENDF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 362 gives a summary of the available features.

Table 362. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit

37.7 PKA registers

37.7.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
										rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
		rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
100000: Montgomery parameter computation then ECC scalar multiplication
100100: ECDSA sign
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE[5:0] bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE[5:0] different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE[5:0] bit is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

0: Disable PKA
1: Enable PKA

When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE[5:0] bitfield for details.

Note: When EN = 0, PKA RAM can still be accessed by the application.

37.7.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY
										r	r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
														r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag

0: No address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE[5:0] bitfield can be used.

1: Only ECDSA verification (MODE[5:0] = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK

This bit is asserted when PKA initialization is complete.

0: PKA is not initialized correctly. START bit cannot be set.

1: PKA is initialized correctly and can be used normally.

37.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.
										w	w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRFC**: Clear operation error flag
- 0: No action
- 1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag
- 0: No action
- 1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
- 0: No action
- 1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
- 0: No action
- 1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

37.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 37.3.3: PKA reset and clocks.

37.7.5 PKA register map

Table 363. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value											0	0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
	Reset value											0	0	0		0	0															0	0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0		0																	

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

38 Public key secure accelerator (PKA)

This section only applies to the STM32G56x (fully) and STM32G55x (only the ECDSA signature verification part) devices.

38.1 PKA introduction

PKA (public key secure accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, the accelerator performs all computations with no further hardware or software elaboration required to process the inputs or the outputs.

For manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.

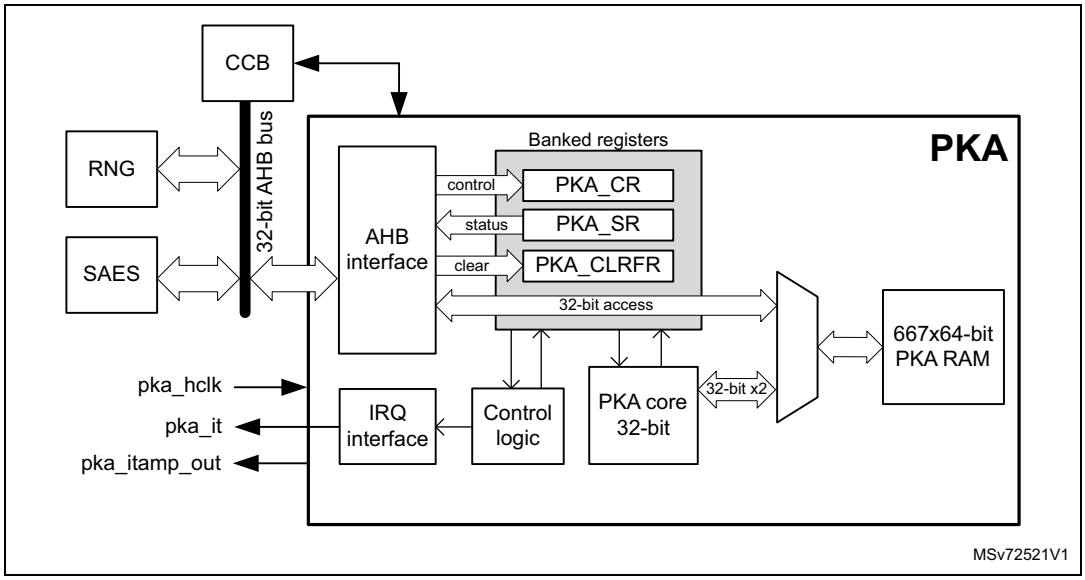
38.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Support for CCB chaining operations required to protect private key used in PKA protected operations
- Hardware protections to monitor usage of private keys during protected operation initialization

38.3 PKA functional description

38.3.1 PKA block diagram

Figure 348. PKA block diagram



38.3.2 PKA internal signals

Table 364 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 364. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key secure accelerator IP global interrupt request
pka_itamp_out	Digital output	PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). A tamper is also generated if application does not respect the maximum supported size of operands, or when PKA finds a fault in critical configuration registers(1). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.

1. When tamper is active, the AHB interface is blocked: RAM and register reads are 0-ed and writes are ignored. There is no PKA RAM erase.

38.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 2000 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to *Section 38.3.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

38.3.4 PKA public key acceleration

Overview

Public key secure accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information can be found in *Section 38.5.1*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 365* and *Table 366*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 365. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R^2 mod *n*	*Section 38.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 38.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 38.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 38.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 38.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	*Section 38.4.6*
0x08	001000	Modular inversion A^{-1} mod n	*Section 38.4.7*
0x0D	001101	Modular reduction A mod n	*Section 38.4.8*
0x09	001001	Arithmetic addition A+B	*Section 38.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 38.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 38.4.11*

Table 365. PKA integer arithmetic functions list (continued)

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 38.4.12*
0x07	000111	RSA CRT exponentiation	*Section 38.4.13*

Table 366. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 38.4.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 38.4.15*
0x23	100011	ECC complete addition	*Section 38.4.18*
0x24	100100	ECDSA sign (protected)	*Section 38.4.16*
0x26	100110	ECDSA verification	*Section 38.4.17*
0x27	100111	ECC double base ladder	*Section 38.4.19*
0x2F	101111	ECC projective to affine	*Section 38.4.20*

Each of these operating modes has an associated code that has to be written to the MODE[5:0] bitfield in the PKA_CR register. If the application selects any value that is not documented below the write to MODE[5:0] bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in *Table 365* and *Table 366* are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

Caution: For security reason it is very important to select protected modular exponentiation (MODE[5:0] = 0x3) when performing RSA decryption.

Montgomery space and fast mode operations

For efficiency reasons, the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming, the application can decide to use a faster mode of operation during which the precomputed Montgomery parameter is supplied before starting the operation. The performance improvement is detailed in *Section 38.5.2: Computation times*.

The only operation using fast mode is the modular exponentiation (MODE[5:0] = 0x02).

38.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions such as:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

The following publications specify the listed functions:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

Refer to these documents for full details. This document limits itself to describing principles.

RSA key pair

For the following RSA operations, a public key and a private key information are defined as follows:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively, this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information, refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*), must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer *m*, with $0 \leq m < n$ and (*m*, *n*) being coprimes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key *d*, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary, retrieve the prime factors (p, q) using (n, e, d) information, then compute phi = (p - 1) × (q - 1). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\,(m_1 - m_2) \bmod p$
   d) $m = m_2 + h\,q$
4. Convert the integer message representative *m* to an encoded message EM.
5. Recover message M = DECODE(EM), where DECODE is a decoding method.

To accelerate these operations with the PKA, use *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

For the decryption process, the protected version of modular exponentiation (MODE[5:0] = 0x3) is strongly recommended for security reason. For the encryption process, the MODE[5:0] = 0x3 setting cannot be used as it requires the knowledge of the private key.

Elliptic curve selection

For the following ECC operations, the curve parameters are defined as follows:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 38.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, $n \times G$ = identity element O).

ECDSA message signature generation

This section describes the ECDSA (elliptic curve digital signature algorithm) signature generation function principle.

Alice, to sign a message *m* using her private key integer d_A, goes through the following steps:

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1}\,(z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

The PKA accelerates the steps 4. to 7. by using:

- *ECDSA sign*, or
- All of the following operations:
  - *ECC Fp scalar multiplication* $k \times P$
  - *Modular reduction* A mod n
  - *Modular inversion* A^{-1} mod n
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

This section describes the ECDSA (elliptic curve digital signature algorithm) signature verification function principle.

Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A. Bob can verify that Q_A is a valid curve point by going through the following steps:

1. Check that Q_A is not equal to the identity element O
2. Check that Q_A is on the agreed curve
3. Check that $n \times Q_A = O$.

Then Bob executes the following procedure:

1. Verify that r and s are integer in [1, n-1]
2. Calculate e = HASH(m), where HASH is the agreed cryptographic hash function
3. Let z be the L_n leftmost bits of e
4. Calculate $w = s^{-1}$ mod n
5. Calculate $u_1 = zw$ mod n and $u_2 = rw$ mod n
6. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. The signature is valid if $r = x_1$ (mod n), it is invalid otherwise.

The PKA accelerates the steps 4. to 7. by using *ECDSA verification*.

38.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit in the PKA_CR register enables the PKA peripheral. The PKA becomes available when the INITOK bit is set in the PKA_SR register. When EN = 0, the PKA peripheral is kept under reset, with its memory still accessible by the application through the AHB interface.

Note: When the PKA is in the process of clearing its memory, the EN bit cannot be set.

When setting the EN bit in PKA_CR, make sure that the MODE[5:0] bitfield value corresponds to an authorized PKA operation (see OPERRF in Section 38.3.8).

Clearing the EN bit while a calculation is in progress aborts the operation. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only the PKA registers can be accessed, with writes to EN bits ignored.

If the INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable the PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 38.4*.

Executing a PKA operation

To execute any of the supported PKA operations, proceed as follows:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE[5:0] bitfield of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1), any access by the application to PKA RAM is ignored, and the RAMERRF flag is set in PKA_SR.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error and prevents the PROCENDF flag from being set and the procedure step *3* from being completed. See *Section 38.3.8* for details.

Using precomputed Montgomery parameters (PKA fast mode)

As explained in *Section 38.3.4*, when computing many operations with the same modulus, it can be beneficial for the application to compute the corresponding Montgomery parameter only once (see, for example, *Section 38.4.5*). This is referred to as *fast mode*.

To use the PKA peripheral in fast mode, follow this recommended procedure:

1. Load the modulus size and value information into the PKA RAM. Refer to *Section 38.5.1* for detail.
2. Configure the PKA_CR register to *Montgomery parameter computation* mode (MODE[5:0] = 0x1), then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set. Then, read the corresponding Montgomery parameter from the PKA memory and clear the PROCENDF bit by setting PROCENDFC in PKA_CLRFR.
4. Continue with the required PKA operation, loading the Montgomery information R2 mod *m* on top of the regular input data. Relevant addresses are provided in *Section 38.4*.

38.3.7 Private keys protection using CCB

Chaining modes

As shown in *Figure 349*, setting CCOP to a value different from 0 in the CCB peripheral automatically activates a chaining mechanism between PKA, RNG, and SAES, depending on the OPSTEP value (see *Table 367*), then sets the CCEN bit in the PKA_SR register.

For the effect of different OPSTEP values in each protected operation (ECDSA signature, scalar multiplication, modular exponentiation), refer to *Section 38.3.7: Private keys protection using CCB*.

Figure 349. Chaining of PKA with SAES or RNG

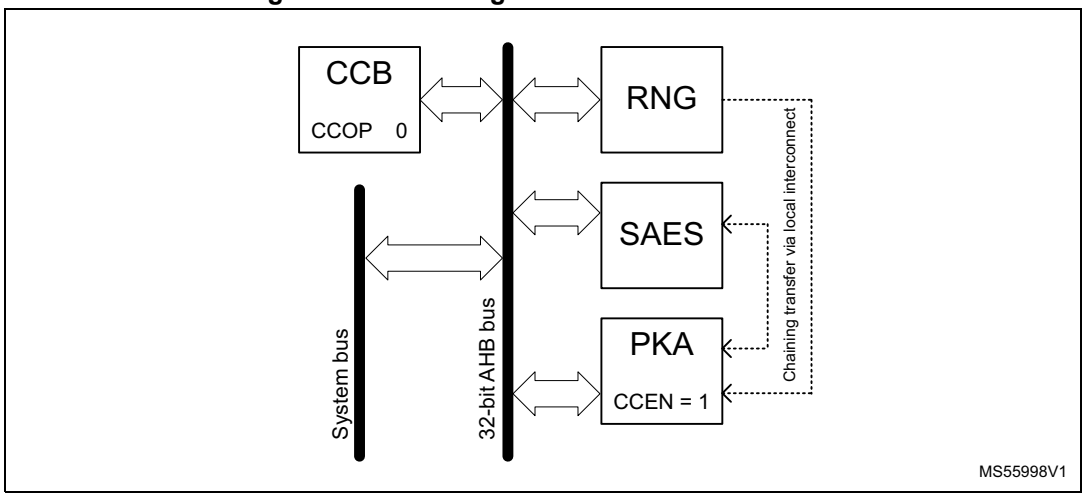


Table 367. Chaining mode to/from PKA

CCOP		OP STEP	Expected MODE[5:0] (1)	Chaining traffic(2)	Protected parameter written to PKA RAM	Expected flags(3)
x	0xC3	0x1	0x24	N/A	Hash (unprotected)	N/A
ECDSA signature	0xC0 or 0xC2	0x3	0x24	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0xC3	0x13				
		0x14		SAES → PKA	Decrypted private key d	DATAOKF or CMF error
		0x16		RNG → PKA	Random integer k	RNGOKF or CMF error
	0xC0	0x4		N/A	Clear text private key d	DATAOKF or CMF error
	0xC2	0x6		RNG → PKA	Random private key d	RNGOKF or CMF error
	0xC0 or 0xC2	0x8		PKA → SAES	Clear-text private key d	DATAOKF or CMF error
Scalar multiplication	0x80 or 0x82	0x3	0x20	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0x81	0x13				
		0x14		SAES → PKA	Decrypted k	DATAOKF or CMF error
	0x80	0x04		N/A	Clear text scalar mult k	
	0x82	0x6		RNG → PKA	Random scalar mult k	RNGOKF or CMF error
	0x80 or 0x82	0x8		PKA → SAES	Clear-text scalar mult k	DATAOKF or CMF error

Table 367. Chaining mode to/from PKA (continued)

CCOP		OP STEP	Expected MODE[5:0] (1)	Chaining traffic(2)	Protected parameter written to PKA RAM	Expected flags(3)
Modular exponentiation	0x44	0x3	0x3	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0x45	0x13				
		0x14		SAES → PKA	Decrypted exponent e	DATAOKF or CMF error
		0x15			Decrypted phi	
	0x44	0x4		N/A	Clear text exponent e	
		0x5			Clear text phi	
		0x8		PKA → SAES	Clear text exponent e	
		0x9			Clear text phi	

1. This mode must be written in PKA_CR before the START bit is set in PKA_CR.
2. Chaining traffic triggered by the CPU access to PKA RAM. “N/A” means no chaining traffic, only CPU traffic.
3. In PKA_SR bits 8 to 15 (“CMF error” refers to error flag bits 8 to 14).
4. In this mode chaining traffic includes the 32-bit data word plus the 32-bit physical address to PKA RAM.
5. Defined in *Table 368*.

When the application uses a PKA protected operation (with its private key), it must load the related blob information in the PKA RAM and in the SAES_DINR register. If the information exactly matches the conditions of the blob generation (the choice and order of writing protected parameters in PKA RAM, the selection of blob encryption key, and the usage of the original encrypted secrets), the CCB allows it to set the START bit in PKA_CR. When the PKA ignores the setting of the START bit, the application must reset the CCB using the IPRST bit in CCB_CR, then start the protected operation sequence again.

A protected application typically uses the PKA in the following sequence composed of three phases.

Step 1: Public phase

Before using a protected operation, the application may need to write unprotected parameters to the PKA RAM, as defined in *Table 367*.

Step 2: Integrity protected phase

To use a PKA operation, the application must load specific parameters in PKA RAM. When CCEN bit is set in PKA_SR, the integrity protected parameters defined in *Table 368* must always be written in the same way. Integrity protection is enforced in specific OPSTEPs, as defined in *Table 367*.

Typical issues with integrity of protected operation parameters are:

- The blob encryption key loaded in SAES registers is not correct.
- At least one confidentiality protected parameter defined in *Table 369* has an incorrect value, or was not properly written (DATAOKF = 0 or RNGOKF = 0, see *PKA status register (PKA_SR)*).
- At least one integrity protected parameter defined in *Table 368* has an incorrect value, is written at an incorrect address, or its write sequence to PKA RAM does not match

the sequence used during the blob creation. On this specific point, the application must follow the sequence detailed in *Table 368* when writing each item described in the column *Parameters*.

CCB feeds integrity protected parameters to SAES during the header phase of AES-GCM (GCMPH = 0x1).

Table 368. Integrity protected parameters list

Protected operation blob(1)	Parameters
ECC key blob	Curve prime order *n* length, curve modulus *p* length, curve coefficient *a* sign, curve coefficient |*a*|, curve coefficient *b*, curve modulus value *p*, curve prime order *n*
ECC signature blob	Same as scalar multiplication operation, adding curve base point G coordinates *x* and *y*
RSA key blob	Exponent *e* length, operand length, modulus value *n*

1. See *Table 369* for more details.

Step 3: Confidentiality phase

To use any PKA protected operation, the application must load confidentiality protected parameters in the PKA RAM. When the CCEN bit is set in PKA_SR, the parameters defined in *Table 369* are protected in confidentiality by:

- Making PKA RAM write-only
- Triggering a MODE[5:0] error (MDERRF) in PKA_SR when MODE[5:0] bitfiled in PKA_CR does not match the expected value when application sets START bit in PKA_CR
- Monitoring the correct management of protected parameters through DATAOKF and RNGOKF flags (see *PKA status register (PKA_SR)*)
- Optionally, generating ECC secrets from the on-chip TRNG when CCOP = 0xC2 or 0x82 and OPSTEP = 0x6, and when CCOP = 0xC3 and OPSTEP = 0x16. The PKA performs a sanity check before accepting this random value (see the RNGERRF flag).

Confidentiality protected parameters are described in *Table 369*. CCB feeds them to SAES during the payload phase of the AES-GCM (GCMPH = 0x2).

Table 369. Confidentiality protected parameter list

PKA operation	Secrets	Protected operation blob(1)	Start offset in PKA RAM	Size(2)
Modular exponentiation (protected, MODE[5:0] = 0x3)	Exponent e	RSA key blob	@0x14B8	ROS
	Phi value		@0xC68	
Scalar multiplication (MODE[5:0] = 0x20)	Scalar k	ECC key blob	@0x12A0	EOS
ECDSA signature (MODE[5:0] = 0x24)	Integer k	Not available	@0x12A0	EOS
	Private key d	ECC signature blob	@0xF28	

1. Information blob where confidentiality and integrity protected parameters are stored (see also *Table 368*)
2. See *Section 38.4.1* for details

Step 4: Running PKA protected operation

Set the START bit in PKA_CR. When the whole sequence executes correctly, the PKA RAM becomes inaccessible for reading and writing. Input parameters are automatically erased when the PROCENDF flag in PKA_SR is set, or if the application aborts the current operation by clearing the EN bit in PKA_CR.

38.3.8 PKA error management

The following error flags in the PKA_SR register are set to indicate errors that may occur when using the PKA:

- **ADDRERRF** flags that the access to PKA RAM falls outside the expected range.
- **RAMERRF** flags an AHB access to the PKA RAM occurring while the PKA core uses the PKA RAM. While the flag is set, reads form the PKA RAM return 0 and writes are ignored.
- **OPERRF** flags that the selected operating mode using MODE[5:0] bitfield is not listed in PKA operating modes (see the bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). While the flag is set, writes to the MODE[5:0] bitfield are ignored.
- An ongoing chaining mode sequence error raises the chaining mode flag **CMF** and the corresponding error flag:
  - **INCRERRF** flags increment error
  - **DATAZF** flags data zeroed error
  - **TRZERRF** flags trailing zeros error
  - **MDERRF** flags mode error
  - **RNGERRF** flags RNG error

  For details on each flag and on enabling the chaining mode sequence, see *Section 38.7.2*.

For each error flag, the PKA generates an interrupt if enabled with the corresponding bit in the PKA_CR register (see *Section 38.6* for details).

The ADDRERRF, OPERRF, and RAMERRF error flags are cleared by setting the ADDRERRFC, OPERRFC, and RAMERRFC bit in PKA_CLRFR, respectively. The OPERRF error flag must be cleared before a new operation is initiated via the PKA_CR register.

Upon an ongoing chaining mode sequence error, the application must set the CMFC bit in the PKA_CLRFR register, then restart the ongoing protected operation sequence in the CCB peripheral.

Note: The PKA can be re-initialized at any moment by clearing the EN bit in the PKA_CR register.

38.4 PKA operating modes

38.4.1 Introduction

The operations supported by the PKA are detailed in the following subsections, which define the format of both input data and results stored in the PKA RAM.

Warning: Before starting any operation, ensure that all input parameters to the PKA are valid and consistent, as the PKA assumes their correctness. Additionally, input parameters must not exceed the operand sizes specified in the operation tables.

The following information applies to all PKA operations:

- The PKA core processes 64-bit words in its RAM. Accordingly, in the PKA section of this document, 'word' denotes a 64-bit unit.
- When writing an element as input in the PKA RAM, append an additional 64-bit word of zeros after the most significant word, except when the operand size is fixed at one word.
- All RAM storage addresses refer to the least significant word of the data. To obtain the actual address, the application must add the base address of the PKA to the indicated offset.
- Supported operand sizes are:
  - **ROS (RSA operand size)**: Data size is calculated as *(rsa_size / 64) + 1* words, where *rsa_size* is the modulus length in bits. For example, a 1024-bit RSA operand corresponds to 17 words (1088 bits).
  - **EOS (ECC operand size)**: Data size is *(ecc_size / 64) + 1* words, where *ecc_size* is the prime modulus length in bits. For example, a 192-bit ECC operand corresponds to four words (256 bits).

  Note: Both ROS and EOS include the additional zero word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Any fractional results in the formulas must be rounded to the nearest integer, as the PKA core processes data in 64-bit words.

Note: The maximum ROS is 66 words (4160-bit maximum exponent size), while the maximum EOS is 11 words (640-bit max operand size).

Example 1: Understanding endianness in PKA memory during ECC Fp scalar multiplication preparation

Write the *x*-coordinate of point P for an ECC P256 curve (EOS = 5 words) by placing the least significant bit at bit 0 of the word at address offset 0x578, and the most significant bit at bit 63 of the word at address offset 0x590. Then, write a zero word (0x0000000000000000) at address offset 0x598.

Example 2: Writing the curve coefficient '*a*' for ECC Fp scalar multiplication

To write the value *a* = -3 for a curve with a modulus length of 224 bits (rounded up to five 64-bit words), write the PKA memory with data as in the following table:

Table 370. Example of *'a'* curve coefficient for ECC Fp scalar

RAM offset	Value	Note
0x410	0x0000000000000001	Curve coefficient *a* sign without extra word
0x418	0x0000000000000011	Value of |*a*| LSB

Table 370. Example of *'a'* curve coefficient for ECC Fp scalar

RAM offset	Value	Note
0x420	0x0000000000000000	-
0x428	0x0000000000000000	-
0x430	0x0000000000000000	Value of |*a*| MSB
0x438	0x0000000000000000	Additional zero word

38.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod n) used by the PKA to convert operands into the Montgomery residue system representation. This operation can be very useful for fast mode operation because passing the Montgomery parameter as input saves the time for its computation.

Note: This operation also supports ECC curves when specifying the prime modulus length and EOS size.

The following table summarizes operation instructions for Montgomery parameter computation.

Table 371. Montgomery parameter computation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	ROS
OUT	Result: R^2 mod *n*	-	RAM@0x620	

38.4.3 Modular addition

The following table summarizes operation instructions for modular addition operation that calculates *A* + *B* mod *n*.

Table 372. Modular addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	(0 ≤ *A* < *n*)	RAM@0xA50	ROS
	Operand *B*	(0 ≤ *B* < *n*)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: *A* + *B* mod *n*	(0 ≤ result < *n*)	RAM@0xE78	

38.4.4 Modular subtraction

Modular subtraction operation calculates:

- ***A* - *B* mod *n***, when $A \geq B$
- ***A* + *n* - *B* mod *n***, when $A < B$

The following table summarizes the operation instructions.

Table 373. Modular subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \leq A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \leq B < n$)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: *A* - *B* mod *n*	($0 \leq$ result $< n$)	RAM@0xE78	

38.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications, the PKA accelerates a multiplication that has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication $A \times B \bmod n$

The method to perform these operations is described next. The *x* function is this operation, and the *A*, *B*, and *C* operands are in the natural domain.

1. Inward or outward conversion to or from the Montgomery domain
   a) Assuming that *A* is an integer in the natural domain:
      – Compute *r2modn* using *Montgomery parameter computation*.
      – The result of $AR = A \times r2modn \bmod n$ is *A* in the Montgomery domain.
   b) Assuming that *BR* is an integer in the Montgomery domain:
      – The result of $B = BR \times 1 \bmod n$ is *B* in the natural domain.
      – Similarly, the *AR* value computed in *a)* can be converted into the natural domain by computing $A = AR \times 1 \bmod n$.
2. Simple modular multiplication $A \times B \bmod n$
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute $AR = A \times r2modn \bmod n$. The output is in the Montgomery domain.
   c) Compute $AB = AR \times B \bmod n$. The output is in the natural domain.
3. Multiple modular multiplication $A \times B \times C \bmod n$
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute $AR = A \times r2modn \bmod n$. The output is in the Montgomery domain.
   c) Compute $BR = B \times r2modn \bmod n$. The output is in the Montgomery domain.
   d) Compute $ABR = AR \times BR \bmod n$. The output is in the Montgomery domain.
   e) Compute $CR = C \times r2modn \bmod n$. The output is in the Montgomery domain.
   f) Compute $ABCR = ABR \times CR \bmod n$. The output is in the Montgomery domain.
   g) (optional) Repeat the two previous steps if multiplying with additional operands is required.
   h) Compute $ABC = ABCR \times 1 \bmod n$ to retrieve the result in the natural domain.

The following table summarizes the operation instructions.

Table 374. Montgomery multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A \times B \bmod n$(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

38.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It calculates of $A^e \bmod n$.

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on *Table 375* and *Table 376*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 377*, for security reason.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE[5:0] = 0x03, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in *Table 375* (normal mode), *Table 376* (fast mode), and *Table 377* (protected mode). Fast mode usage is explained in *Section 38.3.6*.

Table 375. Modular exponentiation (normal mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	

Table 376. Modular exponentiation (fast mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
IN/OUT	Montgomery parameter R^2 mod *n*	(mandatory)	RAM@0x620	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	

Table 377. Modular exponentiation (protected mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x03	PKA_CR	6 bits
	Exponent *e* length	(in bits, not null)	RAM@0x400	64 bits
	Modulus or operand length	(in bits, not null)	RAM@0x408	
	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0x16C8	ROS
	Exponent *e*	($0 \le e < n$)	RAM@0x14B8	
	Modulus value *n*	(odd integer only, $n < 2^{4096}$)	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	
ERROR	Error A^e mod *n*	No errors: 0xD60D Errors: 0xCBC9	RAM@0x1298	64 bits

1. Euler totient function of n[1] with phi = (p - 1) × (q - 1), where p and q are prime factors of modulus n (see NIST SP800-56B Appendix C or *RSA encryption/decryption principle* for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as (p, q, e, d) instead of (N, e, d, *phi*), in this case, to derive N and *phi* using PKA arithmetic multiplier: $N = p \times q$, and *phi* = (p - 1) × (q - 1).

38.4.7 Modular inversion

The modular inversion operation calculates multiplicative inverse A^{-1} mod n. If the modulus n is prime, for all values of A ($1 \le A < n$) modular inversion output is valid. If the modulus n is not prime, A has an inverse only if the greatest common divisor between A and n is 1.

If the operand A is a divisor of the modulus n the result is a multiple of a factor of n.

The following table summarizes the operation instructions.

Table 378. Modular inversion

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result: A^{-1} mod *n*	0 < result < *n*	RAM@0xE78	

38.4.8 Modular reduction

The modular reduction operation calculates the remainder of A divided by n.

The following table summarizes the operation instructions.

Table 379. Modular reduction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand *A*	($0 \le A < 2n < 2^{4160}$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result *A* mod *n*	(0 < result < *n*)	RAM@0xE78	

38.4.9 Arithmetic addition

The arithmetic addition operation calculates *A* + *B*.

The following table summarizes the operation instructions.

Table 380. Arithmetic addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x09	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* + *B*	($0 \le$ result $< 2^{M+1}$)	RAM@0xE78	ROS + 1

38.4.10 Arithmetic subtraction

The arithmetic subtraction operation calculates:

- $\boldsymbol{A - B}$, when $A \ge B$
- $\boldsymbol{A + 2^{int(M / 32) \times 32 + 1} - B}$, when $A < B$ and the residue of $M / 32$ is greater than zero
- $\boldsymbol{A + 2^{int(M / 32) \times 32} - B}$, when $A < B$ and the residue of $M / 32$ is zero

For the last two list items, the 32-bit word following the most significant word of the output is 0xFFFF FFFF, as the result is negative.

The following table summarizes the operation instructions.

Table 381. Arithmetic subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0A	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le < A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le < B < 2^M$)	RAM@0xC68	
OUT	Result: *A* - *B*	($0 \le$ result $< 2^M$)	RAM@0xE78	

38.4.11 Arithmetic multiplication

The arithmetic multiplication operation calculates A × B.

The following table summarizes the operation instructions.

Table 382. Arithmetic multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0B	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* × *B*	($0 \le$ result $< 2^{2M}$)	RAM@0xE78	2xROS

38.4.12 Arithmetic comparison

The arithmetic comparison operation yields the following results:

- **0xED2C**, when *A* = *B*
- **0x7AF8**, when *A* > *B*
- **0x916A**, when *A* < *B*

The following table summarizes the operation instructions.

Table 383. Arithmetic comparison

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0C	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	If result *A* = *B*	0xED2C, 0x7AF8, or 0x916A	RAM@0xE78	64 bits

38.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation m = A^d (mod pq) more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod q$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq \bmod pq$

Operation instructions for computing CRT exponentiation A^d mod pq are summarized in *Table 384*.

Table 384. CRT exponentiation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	$(0 < d_P < 2^{M/2})$	RAM@0x730	ROS / 2
	Operand d_Q	$(0 < d_Q < 2^{M/2})$	RAM@0xE78	
	Operand q_{inv}	$(0 < q_{inv} < 2^{M/2})$	RAM@0x948	
	Prime p(1)	$(0 < p < 2^{M/2})$	RAM@0xB60	
	Prime q(1)	$(0 < q < 2^{M/2})$	RAM@0x1088	
IN	Operand A	$(0 < A < 2^M)$	RAM@0x12A0	ROS
OUT	Result: A^d mod pq	(0 ≤ result < pq)	RAM@0x838	

1. Must be different from 2.

38.4.14 Point on elliptic curve Fp check

This operation checks whether a given point $P(x, y)$ satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where a and b are elements of the curve.

The following table summarizes the operation instructions.

Table 385. Point on elliptic curve *Fp* check

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x28	PKA_CR	6 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(|*b*| < *p*)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, 0 < *p* < 2640)	RAM@0x470	
	Point P coordinate *x*	(*x* < p)	RAM@0x578	
	Point P coordinate *y*	(*y* < *p*)	RAM@0x5D0	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Result: point *P* on curve	0xD60D: point on curve 0xA3B7: point not on curve 0xF946: *x* or *y* coordinate is not smaller than modulus *p*	RAM@0x680	64 bits

38.4.15 ECC Fp scalar multiplication

This operation calculates $k \times P\,(x_P, y_P)$, where *P* is a point on a curve over prime fields and *x* is the elliptic curve scalar point multiplication. The result is a point that belongs to the same curve or a point at infinity.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation *point on elliptic curve* can be used to avoid this.

The following table summarizes the operation instructions.

Table 386. ECC Fp scalar multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x20	PKA_CR	6 bits
IN	Curve prime order *n* length	(in bits, not null,)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	

Table 386. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (note)	Storage	Size
IN	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Scalar multiplier *k*	($0 \le k < n$)	RAM@0x12A0	
	Point P coordinate x_P	($x_P < p$)	RAM@0x578	
	Point P coordinate y_P	($y_P < p$)	RAM@0x470	
	Curve prime order *n*	(integer prime)	RAM@0xF88	
OUT	Result: *k* × *P* coordinate *x'*	(result < *p*)	RAM@0x578	EOS
	Result: *k* × *P* coordinate *y'*	(result < *p*)	RAM@0x5D0	
ERROR	Error *k* × *P*	No errors: 0xD60D Errors: 0xCBC9	RAM@0x680	64 bits

When performing this operation the following special cases must be noted:

- For *k* = 0 this function returns a point at infinity (0, 0) if curve parameter *b* is nonzero, (0, 1) otherwise. For *k* different from 0 it might happen that a point at infinity is returned. When the application detects this behavior, a new computation must be carried out.
- For *k* < 0 (that is, a negative scalar multiplication is required) the multiplier absolute value
  k = |-*k*| must be provided to the PKA. After the computation completion, the formula -*P* = (*x*, -*y*) can be used to compute the *y* coordinate of the effective final result (the *x* coordinate remains the same).

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

38.4.16 ECDSA sign

Table 387 (input parameters) and *Table 388* (output parameters) summarize the ECDSA signing operation.

Upon an output error different from 0xD60D, the application must generate a new *k* and repeat the ECDSA sign operation.

Note: As long as this operation is ongoing (BUSY = 1), the PKA control register and the PKA memory are not accessible. The access is restored when the PKA clears the BUSY bit.

The operation completion with errors due to unexpected hardware events triggers a PKA tamper event in the TAMP peripheral and blocks access to the PKA RAM until the hardware erases the event. Missing programmed input point on the input ECC curve also triggers a PKA tamper event. The use of the *point on elliptic curve* PKA operation prevents this.

Table 387. ECDSA sign - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x24	PKA_CR	6 bits
	Curve prime order *n* length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Integer *k*(1)	($0 \leq k <$ n)	RAM@0x12A0	
	Curve base point *G* coordinate *x*	($x < p$)	RAM@0x578	
	Curve base point *G* coordinate *y*	($y < p$)	RAM@0x470	
	Hash of message *z*	(hash size equal to *nlen*)(2)	RAM@0xFE8	
	Private key *d*	($0 < d < n$)	RAM@0xF28	
	Curve prime order *n*	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases *k* can be deterministically generated.
2. To match the hash parameter size to the length of the curve prime order *n,* pad the result with zeros or use the hash truncation.

Table 388. ECDSA sign - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Signature part *r*	($0 < r < n$)	RAM@0x730	EOS
	Signature part *s*	($0 < s < n$)	RAM@0x788	
ERROR	Result of signature	0xD60D: successful computation, no error 0xCBC9: failed computation 0xA3B7: signature part r is equal to 0 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note: The error code is not affected when the PKA automatically clears the PKA RAM at the end of this protected operation. The error output 0xD60D does not affect the result output either.

Extended ECDSA support

The PKA also supports the extended ECDSA signature, which accepts the same inputs as the standard ECDSA signature (see *Table 387*) and produces similar outputs (see *Table 388*), with the addition of the point *kG* coordinates. The following table defines these additional outputs.

Table 389. Extended ECDSA sign - additional outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Curve point kG coordinate x_1	($0 \le x_1 < p$)	RAM@0x1400	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

38.4.17 ECDSA verification

Table 390 (input parameters) and *Table 391* (output parameters) summarize the ECDSA verification operation.

Upon an output error different from 0xD60D, the signature is not verified.

Table 390. ECDSA verification - inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x26	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus p length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient $\|a\|$	(absolute value, $\|a\| < p$)	RAM@0x470	EOS
	Curve modulus value p	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x4D0	
	Curve base point G coordinate x	($x < p$)	RAM@0x678	
	Curve base point G coordinate y	($y < p$)	RAM@0x6D0	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part r	($0 < r < n$)	RAM@0x10E0	
	Signature part s	($0 < s < n$)	RAM@0xC68	
	Hash of message z	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order n	(integer prime)	RAM@0x1088	

1. To match the hash parameter size to the length of the curve prime order n, pad the result with zeros or use the hash truncation.

Table 391. ECDSA verification - outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– (0 < *r* < *n*)	RAM@0x578	EOS

38.4.18 ECC complete addition

The ECC complete addition operation calculates the sum of two points on an elliptic curve.

Operation instructions are summarized in *Table 392*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. To convert resulting point to Jacobian coordinates conversion (x, y)= (X / Z^2, Y / Z^3) can be used.

Table 392. ECC complete addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	
	First point P coordinate *X*	(*x* < *p*)	RAM@0x628	
	First point P coordinate *Y*	(*y* < *p*)	RAM@0x680	
	First point P coordinate *Z*	(*z* < *p*)	RAM@0x6D8	
	Second point Q coordinate *X*	(*x* < *p*)	RAM@0x730	
	Second point Q coordinate *Y*	(*y* < *p*)	RAM@0x788	
	Second point Q coordinate *Z*	(*z* < *p*)	RAM@0x7E0	
OUT	Result coordinate *X*	(*x* < *p*)	RAM@0xD60	
	Result coordinate *Y*	(*y* < *p*)	RAM@0xDB8	
	Result coordinate *Z*	(*z* < *p*)	RAM@0xE10	

38.4.19 ECC double base ladder

ECC double base ladder operation consists in the computation of $k \times P + m \times Q$, where (P, Q) are two points on an elliptic curve and (k, m) are two scalars. Operation instructions are summarized in *Table 393*.

If the resulting point is the point at infinity (error code 0xA3B7), the resulting coordinate equals to (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 393. ECC double base ladder

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer *k*	($0 < k < 2^{640}$)	RAM@0x520	
	Integer *m*	($0 < m < 2^{640}$)	RAM@0x578	
	First point *P* coordinate *X*	($x < p$)	RAM@0x628	
	First point *P* coordinate *Y*	($y < p$)	RAM@0x680	
	First point *P* coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point *Q* coordinate *X*	($x < p$)	RAM@0x730	
	Second point *Q* coordinate *Y*	($y < p$)	RAM@0x788	
	Second point *Q* coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *x*	($x < p$)	RAM@0x578	EOS
	Result coordinate *y*	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

38.4.20 ECC projective to affine

ECC projective to affine operation calculates the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (*X*, *Y*, *Z*), it calculates the affine coordinates (*x*, *y*) = (*X* / *Z*, *Y* / *Z*).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 394*.

Table 394. ECC projective to affine

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate *X* (projective)	($x < p$)	RAM@0xD60	
	Point P coordinate *Y* (projective)	($y < p$)	RAM@0xDB8	
	Point P coordinate *Z* (projective)	($z < p$)	RAM@0xE10	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Point P coordinate *x* (affine)	($x < p$)	RAM@0x578	
	Point P coordinate *y* (affine)	($y < p$)	RAM@0x5D0	
ERROR	Error code	Point not at infinity: 0xD60D Point at infinity: 0xA3B7	RAM@0x680	64 bits

38.5 Example of configurations and processing times

38.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b$ (mod p).

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field *GF(p)*
- the (usually prime) integer *n*, the order of the group generated by *G*, defined below
- the base point of the curve *G*, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. ***a* defined as *p*-|*a*|** in the finite field *GF(p)*, for example ***p*-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 395 summarizes the family of curves supported by PKA for ECC operations.

Table 395. Family of supported curves for ECC operations

Curve name	Standard	Reference	
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4	
P-224			
P-256			
P-384			
P-521			
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

38.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 396. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	124600	491000	684000	1133200
	Fast	22700	82000	178000	311000
17	Normal	135700	531400	772400	1288000
	Fast	33800	122500	266500	465800
2^{16} + 1	Normal	180000	693700	1126200	1907200
	Fast	78200	284700	620400	1085000
1024	Protected	9958000	-	-	-
	Normal	5850000	-	-	-
	Fast	5748000	-	-	-
	CRT(1)	1775000	-	-	-
2048	Protected	-	63886000	-	-
	Normal	-	42240000	-	-
	Fast	-	41832000	-	-
	CRT(1)	-	11670000	-	-
3072	Protected	-	-	199403000	-
	Normal	-	-	136830000	-
	Fast	-	-	136325000	-
	CRT(1)	-	-	36886000	-
4096	Protected	-	-	-	454318000
	Normal	-	-	-	316000000
	Fast	-	-	-	315226000
	CRT(1)	-	-	-	84577000

1. CRT stands for chinese remainder theorem optimization (MODE[5:0] bitfield= 0x07).

Table 397. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
-	1590000	3083000	5339000	8518000	17818000	21053000	31826000

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 398. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
-	1500000	2744000	4579000	7184000	14455000	16685000	24965000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 399. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
1011000	1495000	2938000	5014000	7979000	16804000	19254000	29582000

Table 400. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
967000	1419000	2768000	4784000	7547000	15854000	18257000	28257000

Table 401. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
47600	78000	148300	253000	419000	838400	1049300

Table 402. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
10000	12000	18000	26000	39000	63000	89000

Table 403. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
3400	4200	6100	8300	10900	17200	-	-

Table 404. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
8600	8710	11870	17000	102000	410000	506000	822000

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

38.6 PKA interrupts

There are five individual maskable interrupt sources generated by the public key secure accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 38.3.8*
2. Access to unmapped address (ADDRERRF), see *Section 38.3.8*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 38.3.8*
4. PKA end of operation (PROCENDF)
5. Chaining mode flags (CMF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

CMF interrupt is triggered by the end of an operation (DATAOKF, RNGOKF) or by a chaining mode error (see *Section 38.3.8*). Chaining modes are described in *Section 38.3.7*.

Table 405 gives a summary of the available features.

Table 405. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit
	Chaining mode flag	CMF	CMFIE	Set CMFC bit

38.7 PKA registers

38.7.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFIE	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
									rw	rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
		rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMFIE**: Chaining mode flags interrupt enable
0: No interrupt is generated when any flag in bits 8 to 15 is set in PKA_SR.
1: An interrupt is generated when any flag in bits 8 to 15 is set in PKA_SR.

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
000011: Modular exponentiation (protected, used when manipulating secrets)
100000: Montgomery parameter computation then ECC scalar multiplication (protected)
100100: ECDSA sign (protected)
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE[5:0] bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE[5:0] different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE[5:0] bit is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

0: Disable PKA

1: Enable PKA; PKA becomes functional when INITOK is set by hardware in PKA_SR. When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE[5:0] bitfield for details.

Note: When EN = 0, PKA RAM can still be accessed by the application.

38.7.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMF	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY
									r	r	r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RNGERRF	MDERRF	TRZERRF	DATAZF	INCRERRF	DATAOKF	RNGOKF	Res.	Res.	Res.	Res.	Res.	CCEN	LMF	INITOK
	r	r	r	r	r	r	r						r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMF**: Chaining mode flags

This status bit is only used when CCEN status bit is set in this register.

0: No chaining mode events.

1: Ongoing chaining mode sequence triggered at least one flag status in bits 8 to 14 of this register.

This bit is cleared using CMFC bit in PKA_CLRFR.

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag

0: No address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

When CCEN = 1 PKA sets BUSY bit when it runs a sanity check on a random number that CCB wrote to the PKA RAM during a chaining sequence (see *Section 38.3.7*).

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RNGERRF**: RNG error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No random number drawn from RNG, or generated ECC private key is NIST compliant.

1: Random number drawn from RNG did not pass NIST “key pair generation by testing candidates” equivalent algorithm (see FIPS 186-4 annex B.4.2). The last chaining operation must be restarted, otherwise the selected PKA operation remains unusable.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 13 **MDERRF**: Mode error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No mode error

1: The value written in MODE[5:0] of PKA_CR registers is different from the expected value corresponding to the CCOP mode selected in the CCB peripheral. Refer to *Section 38.3.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 12 **TRZERRF**: Trailing zeroes error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No trailing 0s error

1: Clear-text coming from CPU, or decrypted data coming from SAES does not have the required trailing 0s.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 11 **DATAZF**: Data zeroed error flag

This read-only chaining mode error flag is active only when CCEN is set in this register. DATAZF is set when PKA finds the complete data written to PKA RAM equal to 0, with all the expected 32-bit words null. This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 10 **INCRERRF**: Increment error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No increment error

1: During the reading or writing of protected parameters from or to the PKA RAM, the application did not properly incremented the address. Refer to *Section 38.3.7* for details.

During the read or write of an operand, and before DATAOKF or RNGOKF is set, the INCRERRF flag stays cleared if the application reads or writes again this operand (starting from the lowest address). This bit is cleared by setting the CMFC bit in PKA_CLRFR.

Bit 9 **DATAOKF**: Data OK flag

This read-only chaining mode status flag is active only when CCEN is set in this register.

0: Data not OK (if applicable), otherwise no event.

1: Sequential writes or reads to or from the PKA RAM successfully transfered the expected amount of data from or to SAES. Alternatively, the CPU wrote the correct amount of data in the PKA RAM. Refer to *Section 38.3.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 8 **RNGOKF**: RNG OK flag

This read-only chaining mode status flag is active only when CCEN is set in this register.

0: Random number not OK (if applicable). No event otherwise.

1: Sequential writes to PKA RAM successfully transfered the expected amount of random data from the RNG. Refer to *Section 38.3.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **CCEN**: Coupling and chaining mode enable

This bit is set when the application configures the CCB to run coupling and chaining sequences that perform protected PKA operation using the SAES and (optionally) RNG peripherals.

0: Access to PKA RAM and programming of PKA_CR has no extra limitation.

1: The PKA RAM is write-only until START is set (it is no access then), and the MODE[5:0] value written in PKA_CR must correspond to the CCOP mode selected in the CCB peripheral. The application must also follow the coupling and chaining sequences described in *Section 38.3.7*.

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE[5:0] bitfield can be used.

1: Only ECDSA verification (MODE[5:0] = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK

This bit is asserted when PKA initialization is complete.

0: PKA is not initialized correctly. START bit cannot be set.

1: PKA is initialized correctly and can be used normally.

When RNG is not able to output proper random numbers or when SAES is using the RNG bus, INITOK remains at 0. If it happens, make sure the RNG peripheral is properly clocked and initialized, and the SAES peripheral is clocked with BUSY bit cleared in SAES_SR to release the RNG bus, then enable PKA again.

38.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFC	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.
									w	w	w	w		w	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMFC**: Clear chaining mode flag

0: No action

1: Clear the CMF flag and the chaining mode status bits in 8 to 15 in PKA_SR

Bit 21 **OPERRFC**: Clear operation error flag

0: No action

1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag

0: No action

1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag

0: No action

1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

38.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 38.3.3: PKA reset and clocks.

38.7.5 PKA register map

Table 406. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFIE	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value										0	0	0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMF	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	RNGERRF	MDERRF	TRZERRF	DATAZF	INCRERRF	DATAOKF	RNGOKF		Res.	Res.	Res.	Res.	Res.	CCEN	LMF	INITOK
	Reset value										0	0	0	0		0	0		0	0	0	0	0	0	0						0	0	0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFC	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0	0		0																	

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

39 On-the-fly decryption engine (OTFDEC)

This section only applies to STM32G56x devices.

39.1 OTFDEC introduction

OTFDEC allows on-the-fly decryption of the AHB traffic based on the read request address information. Four independent and non-overlapping encrypted regions can be defined in OTFDEC.

OTFDEC uses AES-128 in counter mode to achieve the lowest possible latency. As a consequence, each time the content of one encrypted region is changed, the entire region must be re-encrypted with a different cryptographic context (key or initialization vector). This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR flash.

Note: When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to access the flash memory using the memory-mapped mode of the flash memory controller.

When security is enabled in the product, OTFDEC can be programmed only by a secure host.

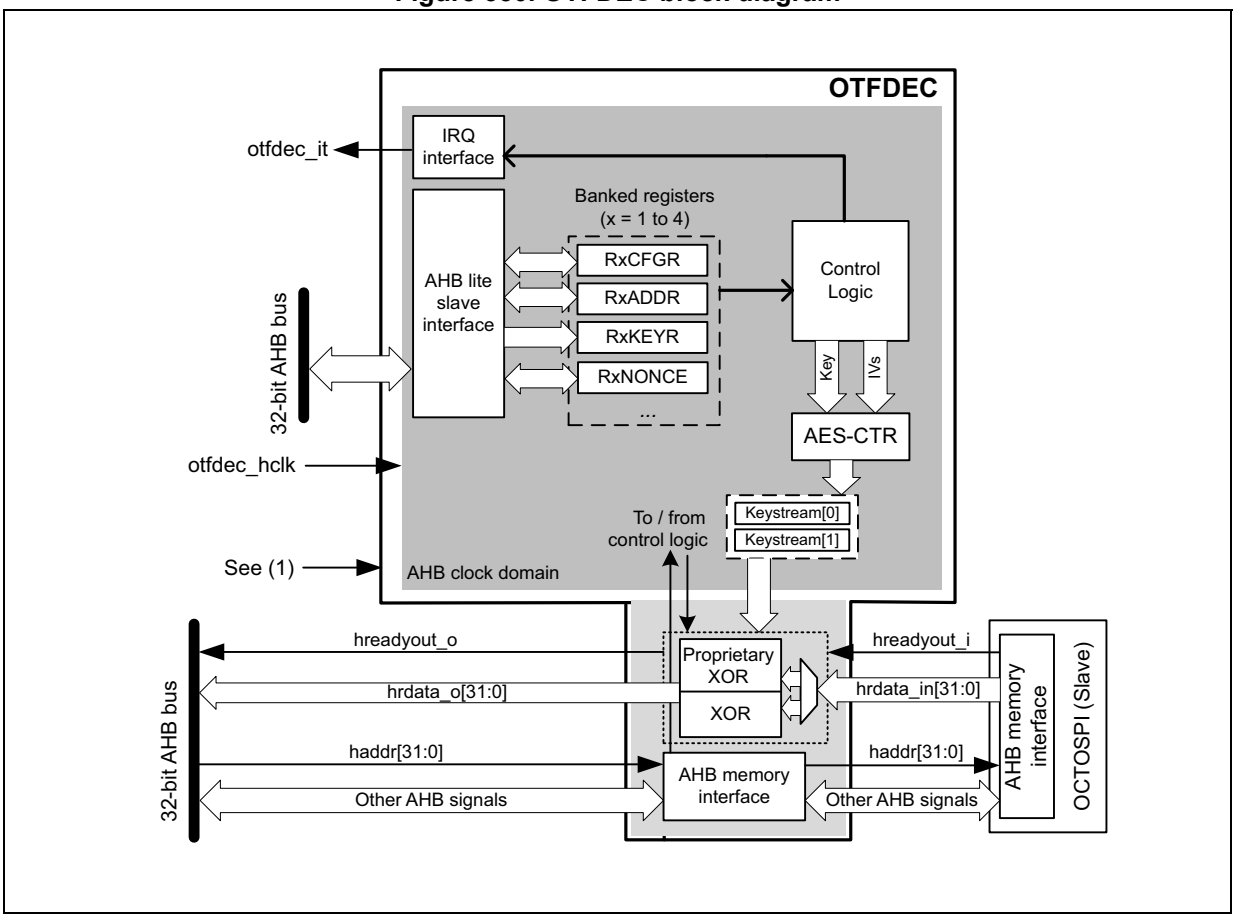
39.2 OTFDEC main features

- On-the-fly 128-bit decryption during the OCTOSPI Memory-mapped read operations (single or multiple).
  - Use of AES in counter (CTR) mode, with two 128-bit keystream buffers
  - Support for any read size
  - Physical address of the reads used for the encryption/decryption
- Up to four independent encrypted regions
  - Granularity of the region definition: 4096 bytes
  - Region configuration write-locking mechanism
  - Each region has its own 128-bit key, two bytes firmware version, and eight bytes application-defined nonce. At least one of those must be changed each time an encryption is performed by the application.
- Encryption keys confidentiality and integrity protection
  - Write-only registers, with software locking mechanism
  - Availability of 8-bit CRC as public key information
- Support for OCTOSPI pre-fetching mechanism
- Possibility to select an enhanced encryption mode to add a proprietary layer of protection on top of AES stream cipher (execute only)
- Privileged-aware AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Secure only programming if TrustZone security is enabled in the product
- Encryption mode

39.3 OTFDEC functional description

39.3.1 OTFDEC block diagram

Figure 350. OTFDEC block diagram



1. otfdec_tzen

39.3.2 OTFDEC internal signals

Table 407 describes a list of useful to know internal signals available at OTFDEC level, not at the product level (on pads).

Table 407. OTFDEC internal input/output signals

Signal name	Signal type	Description
otfdec_hclk	Digital input	AHB bus clock
otfdec_it	Digital output	OTFDEC global interrupt request
otfdec_tzen	Digital input	OTFDEC TrustZone enable, controlling TrustZone features of the peripheral (TZEN)

The TZEN option bit in FLASH is used to activate TrustZone in the device.

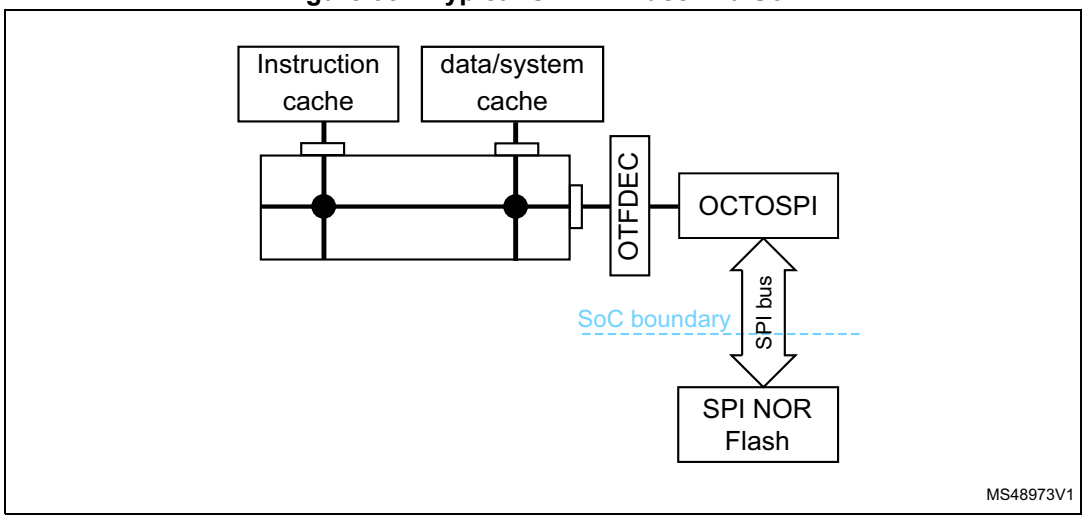
- TZEN = 1: TrustZone security is enabled in the product.
- TZEN = 0: TrustZone security is disabled in the product.

39.3.3 OTFDEC on-the-fly decryption

Introduction

Typical usage for OTFDEC is shown on *Figure 351*.

Figure 351. Typical OTFDEC use in a SoC



Original purpose of OTFDEC is to protect the confidentiality of read-only firmware libraries stored in external SPI NOR flash devices.

A special locking scheme is available in OTFDEC in order to protect the integrity of the decryption keys and also to protect the other configurations against software denial of services attacks. OTFDEC access to most registers can be made privileged-only by setting PRIV bit in OTFDEC_PRIVCFGR register. OTFDEC is only writeable by TrustZone CPU, when TrustZone security is activated.

When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to read the flash memory using the Memory-mapped mode of the flash controller.

On top of decrypting on-the-fly, OTFDEC can also encrypt 32-bit word at a time (see *Section 39.5.3: Encrypting for OTFDEC* for more details).

OTFDEC architecture

OTFDEC analyzes all AHB read transfers on the associated AHB bus. If the read request is within one of the four regions programmed in OTFDEC, the control logic triggers a keystream computation based on AES algorithm in counter mode. This keystream is then used to decrypt on-the-fly the data present in the read transfer from the OCTOSPI AHB master, tying low the HREADYOUT signal of this master while the keystream information is being computed (this takes up to 11 cycles). Any accesses outside the enabled OTFDEC regions belong to a non-encrypted region.

Each OTFDEC region is programmed through OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR, OTFDEC_RxNONCER and

OTFDEC_RxKEYR registers, where x = 1 to 4. In OTFDEC_RxCFGR, the MODE bits define the OTFDEC operating mode (standard or enhanced encryption).

Granularity for the region determination is 4096 bytes.

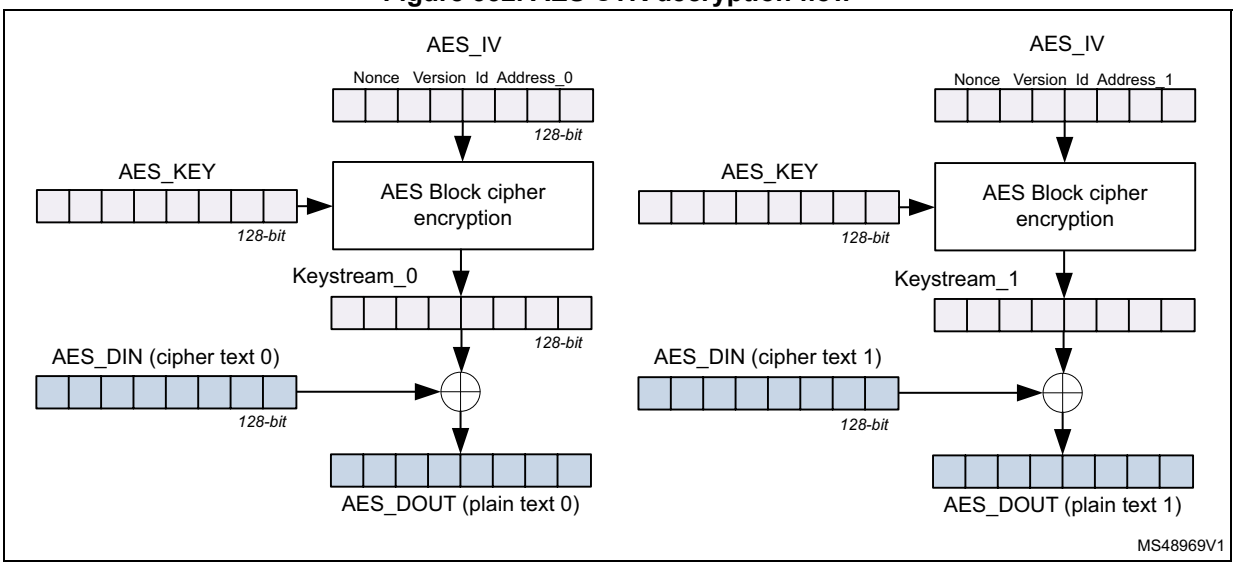
Note: Although OTFDEC does not prevent region overlapping, it is not a valid programming and it must be avoided by application software.

OTFDEC can decrypt incremental or wrap bursts only if they do not cross the 4096-byte aligned address boundaries.

39.3.4 OTFDEC usage of AES in counter mode decryption

Figure 352 shows how OTFDEC uses industry standard Advanced Encryption Standard (AES) algorithm in counter chaining mode. This mode is specified by NIST in *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Figure 352. AES CTR decryption flow



Every 128-bit data block, a special keystream information is computed using AES block cipher, as defined below:

- initialization vector AES_IV[127:0] = RxNONCER1[31:0] || RxNONCER0[31:0] || 0b0000 0000 0000 0000 || RxCFGR[31:16] || 0b00 || (x-1) || ReadAddress[31:4]
- key material AES_KEY[127:0] = R*x*KEYR3[31:0] || R*x*KEYR2[31:0] || R*x*KEYR1[31:0] || R*x*KEYR0[31:0]

Note: Above x is the RegionID of the selected encrypted region (x=1 to 4).
ReadAddress is the AHB address of the encrypted data block, modulo 128-bit.

Resulting 128-bit keystream is XORed with 128-bit cipher text data to produce the 128-bit clear text data.

- AES_DIN and AES_DOUT data blocks are constructed following the rule below (“|” represents a binary concatenation):
  AES_Dx[127:0]= AHB_word(@ | 0xC)[31:0] | AHB_word(@ | 0x8)[31:0] | AHB_word(@ | 0x4)[31:0] || AHB_word(@ | 0x0)[31:0], where @ is the hexadecimal address used to compute the keystream (ReadAddress[31:4] above).

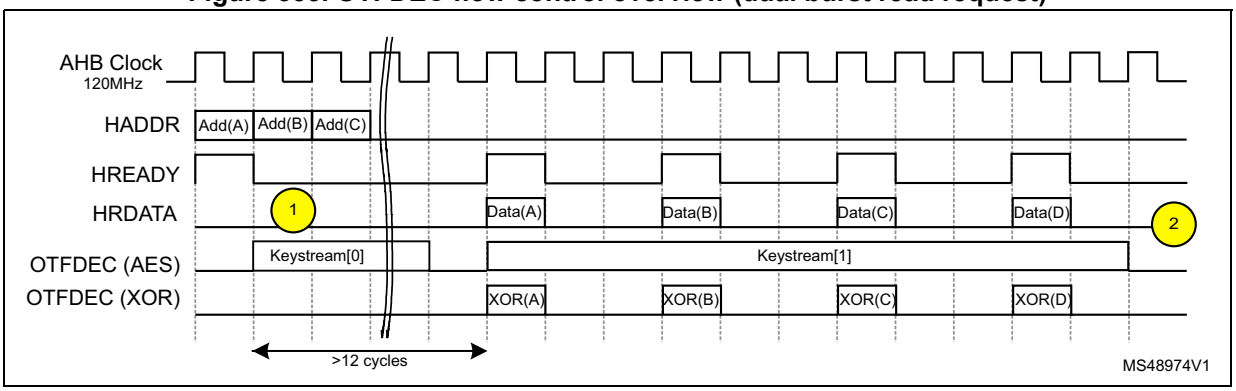
When the read request is not within an encrypted region, or the decryption is not enabled in this region, the AHB data is not changed.

Note: When the application sets the MODE bitfield to 11 in OTFDEC_RxCFGR, an additional layer of protection is added on top of the AES stream cipher. This enhanced encryption mode can only be used with instructions (execute-only region).

39.3.5 Flow control management

Figure 353 shows how OTFDEC manages one INCR4 AHB burst that corresponds to one 128-bit AES data block.

Figure 353. OTFDEC flow control overview (dual burst read request)



with the following notes:

1. OTFDEC enforces HREADY signal from the AHB master low as it is not ready to decrypt data (keystream computation).
2. Thanks to the keystream buffer, OTFDEC can be ready to process a new batch of data within 12 cycles in this configuration (120 MHz AHB clock, 104 MHz SPI bus delivering 2 bytes per SPI clock).

39.3.6 OTFDEC error management

OTFDEC automatically manages errors defined as below:

- Illegal read to OTFDEC_RxKEYR registers
- Illegal write to OTFDEC_RxKEYR registers while CONFIGLOCK or KEYLOCK = 1 in OTFDEC_RxCFGR, while the access is secure. If the security is disabled in the product, the same error occurs when the access is nonsecure.
- Illegal write to OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR or OTFDEC_RxNONCER registers while CONFIGLOCK = 1 in OTFDEC_RxCFGR (x = 1 to 4), while the access is secure. If the security is disabled in the product the same error occurs when the access is nonsecure.
- Illegal read to an execute-only region (MODE[1:0] = 11). Such illegal request returns 0x0, without bus error.
- Execution request to a region while encryption is enabled (ENC = 1). The request returns 0x0, without bus error.
- Key error: read request to an encrypted region while its key registers are null or not properly initialized (KEYCRC = 0x0). Source of the error can be an incorrect key loading sequence (see KEYCRC in OTFDEC_RxCFGR) or it can be an abort event

(tamper detection, unauthorized debug connection, untrusted boot, RDP level regression). Such read request returns 0x0, without bus error.

- Write to any registers while the access is nonsecure, if TrustZone security is enabled in the product.

  This last error is managed and cleared through TrustZone interrupt controller, as described in the GTZC section of the product reference manual.

For these errors (except the last one), an interrupt can be generated if the SEIE, XONEIE or KEIE bit is set in OTFDEC_IER register (see *Section 39.4*).

Note: After a key error, OTFDEC keys must be properly initialized again, and a reset of OTFDEC may be needed if registers are locked.

39.4 OTFDEC interrupts

There are three independent maskable interrupt sources generated by the OTFDEC, signaling following security events:

- Illegal read or write access to keys (SEIF flag), see *Section 39.3.6*
- Illegal write to a region configuration while CONFIGLOCK = 1 (SEIF flag), see *Section 39.3.6*
- Read access to an execute-only region (MODE[1:0] = 11), triggering the XONEIF flag
- Executing while encryption is enabled (XONEIF flag)
- Key error (encrypted regions read as zero) triggering the KEIF flag, see *Section 39.3.6*.

Interrupt sources are connected to the same global interrupt request signal.

OTFDEC interrupt sources can be enabled/disabled by setting the corresponding SEIE, XONEIE or KEIE bit in OTFDEC_IER, as described in *Table 408*. Status of the interrupt event is found in OTFDEC_ISR, and this event can be cleared using OTFDEC_ICR.

Table 408. OTFDEC interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method
OTFDEC	Security error	SEIF	SEIE	Set SEIF in OTFDEC_ICR
	Execute-only Execute while encryption	XONEIF	XONEIE	Set XONEIF in OTFDEC_ICR
	Key error	KEIF	KEIE	Set KEIF in OTFDEC_ICR

1. The event flags are found in the OTFDEC_ISR register.

39.5 OTFDEC application information

39.5.1 OTFDEC initialization process

Introduction

One key aspect of OTFDEC is the trusted initialization of its registers, as it involves secret keys. Two trusted initialization schemes are recommended here below.

Note: Those sequences are for production code, as during firmware development, it is not always recommended to lock the key or the region configuration.

Writes to configuration registers are effective when the configuration locks allow it, even if the region is enabled.

Initialization scheme 1: one key for all regions

In this scheme, one entity owns the secret key used to decrypt the four protected regions. The recommended OTFDEC configuration sequence is described below:

1. For *x* = 1 to 4, write the correct MODE[1:0] value in OTFDEC_RxCFGR.
2. For *x* = 1 to 4, program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
3. For *x* = 1 to 4, check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers in this region *x* are no more writable).
4. To do to decrypt a region x (task that does not necessarily have to be performed by the entity that owns the decryption keys):
   a) Verify if the key CRC corresponds to the encrypted binary stored in the region.
   b) Fill the detailed information corresponding to this binary (nonce, start address, end address, version number).
   c) Enable decryption of this region using REG_EN.
   d) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence, step 1 above must be done before step 2, and MODE bits must not be modified after step 2.

Initialization scheme 2: one key per region

In this scheme, one entity can own the secret used to decrypt one (or more) protected region. The recommended OTFDEC configuration sequence is described below:

1. To do to decrypt a region x (this task **must** be performed by the entity that owns the corresponding key):
   a) Write the correct MODE[1:0] value in OTFDEC_R*x*CFGR.
   b) Program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
   c) Check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers are no more writable).
   d) Fill the detailed information corresponding to the protected firmware (nonce, start address, end address, version number).
   e) Enable decryption of this region using REG_EN.
   f) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence step a) above must be done before step b), and MODE bits must not be modified after step b).

39.5.2 OTFDEC and power management

Each time OTFDEC is reset, the correct key loading sequence described in *Section 39.5.1* must be performed (in this case KEYCRC = 0 in OTFDEC_RxCFGR).

It is recommended for application software to verify this point each time OTFDEC is reset by hardware.

39.5.3 Encrypting for OTFDEC

Code and data standard encryption

OTFDEC uses standard AES in counter mode when processing a binary stored in a protected region with MODE[1:0] = 10. When this mode is selected, any AES compatible hardware accelerator or library can be used to encrypt those protected libraries. OTFDEC can be used as well, as described in enhanced encryption section below (with MODE[1:0] = 10).

Definition and endianness of the AES inputs and outputs are defined in *Section 39.3.4: OTFDEC usage of AES in counter mode decryption*.

Enhanced encryption with OTFDEC

OTFDEC uses a proprietary layer of protection on top of the standard AES in counter mode when processing a code stored in a protected region with MODE[1:0] = 11.

Enhanced encryption mode can be used to increase the robustness against tampering.

Recommended sequence to encrypt using OTFDEC is described below:

1. The application in charge of the encryption sets the ENC bit in OTFDEC_CR. This application must run in TrustZone secure mode when TrustZone security is enabled in the product. If PRIV bit is set in OTFDEC_PRIVCFGR, this application must be privileged.
2. Encryption application initializes OTFDEC as described in *Section 39.5.1: OTFDEC initialization process*. OCTOSPI must also be properly clocked, so that OTFDEC is fully functional in encryption mode. This step can also be done before step 1.
3. Encryption application writes 32-bit of clear-text data at the expected protected address, then reads it back encrypted at the same address to store it in RAM. Note that this data stays inside the device, as it is intercepted by OTFDEC in encryption mode.
4. Encryption application goes back to previous step (changing the address) until the whole binary is processed.
5. Encryption application clears the ENC bit in OTFDEC_CR. Another application can then take the encrypted binary and flash it to the correct address in external flash.

There are few important notes about this procedure:

- Encryption granularity is 32-bit (single 32-bit access is mandatory).
- While ENC bit is set, reads to non-encrypted regions return normal data (such as no encryption nor decryption). While in encryption mode, no access to OCTOSPI (including registers) must be done. This is because the OTFDEC cuts the communication with OCTOPSI while ENC bit is set.
- OTFDEC does not support execution while ENC = 1 (only encrypted data reads). Upon illegal execution detection a XONEIF flag is raised and zero is returned.

39.5.4 OTFDEC key CRC source code

Below is the CRC source code that can be used to compare with the result of the computation provided by OTFDEC in KEYCRC bitfield after loading the keys in OTFDEC_RxKEYR registers.

```
uint8_t getCRC(uint32_t * keyin)
{
  const uint8_t CRC7_POLY = 0x7;
  const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
  uint8_t  i, j, k, crc = 0x0;
  uint32_t  keyval;

  for (j = 0; j < 4; j++)
  {
    keyval = *(keyin+j);
    if (j == 0)
    {
      keyval ^= key_strobe[0];
    }
    else
    {
      keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8)
| crc;
    }

    for (i = 0, crc = 0; i < 32; i++)
    {
      k = (((crc >> 7) ^ (keyval >> (31-i))&0xF)) & 1;
      crc <<= 1;
      if (k)
      {
        crc ^= CRC7_POLY;
       }
    }
    crc^=0x55;
  }
  return crc;
}
```

39.6 OTFDEC registers

39.6.1 OTFDEC control register (OTFDEC_CR)

Address offset: 0x0

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENC**: Encryption mode bit

When this bit is set, OTFDEC is used in encryption mode, during which application can write clear text data then read back encrypted data. When this bit is cleared (default), OTFDEC is used in decryption mode, during which application only read back decrypted data. For both modes, cryptographic context (keys, nonces, firmware versions) must be properly initialized. When this bit is set, only data accesses are allowed (zeros are returned otherwise, and XONEIF is set). When MODE = 11, enhanced encryption mode is automatically selected.

0: OTFDEC working in decryption mode

1: OTFDEC working in encryption mode

Note: When ENC bit is set, no access to OCTOSPI must be done (registers and Memory-mapped region).

39.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR)

Address offset: 0x10

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PRIV:** Privileged access protection.

0: No additional protection is added on OTFDEC register accesses.

1: An additional protection is added when accessing all registers except OTFDEC_PRIVCFGR:

- Unprivileged read accesses to registers return zeros
- Unprivileged write accesses to registers are ignored.

Note: This bit can only be written in privileged mode. There is no limitations on reads.

39.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR)

Address offset: 0x20 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

Writes are ignored if CONFIGLOCK bit is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_VERSION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYCRC[7:0]								Res.	Res.	MODE[1:0]		Res.	KEYLOCK	CONFIGLOCK	REG_EN
r	r	r	r	r	r	r	r			rw	rw		rs	rs	rw

Bits 31:16 **REG_VERSION[15:0]**: region firmware version

This 16-bit bitfield must be correctly initialized before the region corresponding REG_EN bit is set in OTFDEC_RxCFGR.

Bits 15:8 **KEYCRC[7:0]**: region key 8-bit CRC

When KEYLOCK = 0, KEYCRC bitfield is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new computation starts as soon as a new valid sequence is initiated, and KEYCRC is read as zero until a valid sequence is completed.

When KEYLOCK = 1, KEYCRC remains unchanged until the next reset.

CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $X^8 + X^2 + X + 1$ (according the convention). Source code is available in *Section 39.5.4*.

This field is read only.

Note: CRC information is updated only after the last bit of the key has been written.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MODE[1:0]**: operating mode

This bitfield selects the OTFDEC operating mode for this region:

10: All read accesses are decrypted (instruction or data).

11: Enhanced encryption mode is activated, and only instruction accesses are decrypted

Others: Reserved

When MODE ≠ 11, the standard AES encryption mode is activated.

When either of the MODE bits are changed, the region key and associated CRC are zeroed.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **KEYLOCK**: region key lock

0: Writes to this region KEYRx registers are allowed.

1: Writes to this region KEYRx registers are ignored until next OTFDEC reset. KEYCRC bitfield is locked.

Note: This bit is set once: if this bit is set, it can only be reset to 0 if the OTFDEC is reset.

Bit 1 **CONFIGLOCK**: region config lock

0: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are allowed.

1: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are ignored until next OTFDEC reset.

Note: This bit is set once. If this bit is set, it can only be reset to 0 if OTFDEC is reset. Setting this bit forces KEYLOCK bit to 1.

Bit 0 **REG_EN**: region on-the-fly decryption enable

0: On-the-fly decryption is disabled for this region.

1: On-the-fly decryption is enabled for this region. Data are XORed with the corresponding keystream.

Note: Garbage is decrypted if region context (version, key, nonce) is not valid when this bit is set.

39.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR)

Address offset: 0x24 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_START_ADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_START_ADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_START_ADDR[31:0]**: Region AHB start address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR register is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits and the 12 LSB bits return zeros .

39.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR)

Address offset: 0x28 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0FFF

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_END_ADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_END_ADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_END_ADDR[31:0]**: Region AHB end address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set, and OTFDEC_RxENDADDR must be strictly greater than OTFDEC_RxSTARTADDR to be valid.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in OTFDEC_RxCFGR is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits return zeros and the 12 LSB bits return ones.

39.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0)

Address offset: 0x2C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_NONCE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_NONCE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_NONCE[31:0]**: Region nonce, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Writing is discarded in this register if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR is set.

39.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1)

Address offset: 0x30 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_NONCE[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_NONCE[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_NONCE[63:32]**: Region nonce, bits [63:32]

Refer to the OTFDEC_RxNONCER0 register for description of the NONCE[63:0] bitfield.

39.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0)

Address offset: 0x34 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[31:0]**: Region key, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Reading this register returns a zero value. Writing to this register is discarded if performed while the region CONFIGLOCK or KEYLOCK bit is set in the OTFDEC_RxCFGR.

Note: When application successfully changes MODE bits in OTFDEC*_RxCFGR and* OTFDEC*_RxKEYR, and associated KEYCRC are erased.*

39.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1)

Address offset: 0x38 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[63:32]**: Region key, bits [63:32]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

39.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2)

Address offset: 0x3C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[95:64]**: Region key, bits [95:64]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

39.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3)

Address offset: 0x40 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[127:96]**: Region key, bits [127:96]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

39.6.12 OTFDEC interrupt status register (OTFDEC_ISR)

Address offset: 0x300

Reset value: 0x0000 0000

Unprivileged reads return zero if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag status

This bit is set by hardware and read only by application. The bit is set when a read access occurs on an encrypted region, while its key registers is null or not properly initialized (KEYCRC = 0x0).

This bit is cleared when the application sets in OTFDEC_ICR the corresponding bit to 1.

0: OTFDEC operates properly.

1: Read access detected on an enabled encrypted region with its key registers null or not properly initialized (KEYCRC = 0x0). OTFDEC returns a zeroed value for the read, and an optional interrupt is generated if bit KEIE is set to 1 in OTFDEC_IER.

After KEIF is set any subsequent read to the region with bad key registers returns a zeroed value. This state remains until those key registers are properly initialized (KEYCRC not zero).

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when a read access and not an instruction fetch is detected on any encrypted region with MODE bits set to 11. Lastly, XONEIF is also set when an execute access is detected while encryption mode is enabled.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No execute-only error status. No interrupt pending.

1: Read access detected on one region with MODE bits set to 11 or execute access detected while ENC = 1. OTFDEC returns a zeroed value for the illegal access, and an optional interrupt is generated if bit XONEIE is set to 1 in OTFDEC_IER.

Bit 0 **SEIF:** Security error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when at least one security error has been detected.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No security error status. No interrupt pending.

1: Security error flag status, with interrupt pending. Actual interrupt generation is dependent on OTFDEC_IER corresponding bit SEIE.

39.6.13 OTFDEC interrupt clear register (OTFDEC_ICR)

Address offset: 0x304

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear
This bit is written by application, and always read as 0.
0: KEIF flag status is not affected.
1: KEIF flag status is cleared in OTFDEC_ISR.
Note: Clearing KEIF does not solve the source of the problem (bad key registers). To be able to access again any encrypted region, OTFDEC key registers must be properly initialized again.

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag clear
This bit is written by application, and always read as 0.
0: XONEIF flag status is not affected.
1: XONEIF flag status is cleared in OTFDEC_ISR.

Bit 0 **SEIF:** Security error interrupt flag clear
This bit is written by application, and always read as 0.
0: SEIF flag status is not affected.
1: SEIF flag status is cleared in OTFDEC_ISR.

39.6.14 OTFDEC interrupt enable register (OTFDEC_IER)

Address offset: 0x308

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	XONEIE	SEIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when KEIF flag status is set.

0: Interrupt generation on key error flag KEIF is disabled (masked).

1: Interrupt generation on key error flag KEIF is enabled (not masked).

Bit 1 **XONEIE:** Execute-only execute-never error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when XONEIF flag status is set.

0: Interrupt generation on execute-only error XONEIF is disabled (masked).

1: Interrupt generation on execute-only error XONEIF is enabled (not masked).

Bit 0 **SEIE:** Security error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when SEIF flag status is set.

0: Interrupt generation on security error SEIF is disabled (masked).

1: Interrupt generation on security error SEIF is enabled (not masked).

39.6.15 OTFDEC register map

Table 409. OTFDEC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	OTFDEC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC
	Reset value																																0
0x04-0x0C	Reserved	Reserved																															
0x10	OTFDEC_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV
	Reset value																																0
0x14-0x1C	Reserved	Reserved																															
0x20	OTFDEC_R1CFGR1	REG1_VERSION[15:0]																KEYCRC[7:0]								Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x24	OTFDEC_R1STARTADDR	REG1_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	OTFDEC_R1ENDADDR	REG1_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	OTFDEC_R1NONCER0	REG1_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	OTFDEC_R1NONCER1	REG1_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	OTFDEC_R1KEYR0	REG1_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 409. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38	OTFDEC_ R1KEYR1	REG1_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	OTFDEC_ R1KEYR2	REG1_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	OTFDEC_ R1KEYR3	REG1_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44 - 0x4C	Reserved	Reserved																															
0x50	OTFDEC_ R2CFGR	REG2_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x54	OTFDEC_ R2STARTADDR	REG2_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	OTFDEC_ R2ENDADDR	REG2_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	OTFDEC_ R2NONCER0	REG2_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	OTFDEC_ R2NONCER1	REG2_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	OTFDEC_ R2KEYR0	REG2_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	OTFDEC_ R2KEYR1	REG2_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	OTFDEC_ R2KEYR2	REG2_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	OTFDEC_ R2KEYR3	REG2_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74 - 0x7C	Reserved	Reserved																															
0x80	OTFDEC_ R3CFGR	REG3_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x84	OTFDEC_ R3STARTADDR	REG3_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	OTFDEC_ R3ENDADDR	REG3_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 409. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8C	OTFDEC_ R3NONCER0	REG3_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	OTFDEC_ R3NONCER1	REG3_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	OTFDEC_ R3KEYR0	REG3_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	OTFDEC_ R3KEYR1	REG3_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C	OTFDEC_ R3KEYR2	REG3_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0	OTFDEC_ R3KEYR3	REG3_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4 - 0xAC	Reserved	Reserved																															
0xB0	OTFDEC_ R4CFGR	REG4_VERSION[15:0]																KEYCRC[7:0]								Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	
0xB4	OTFDEC_ R4STARTADDR	REG4_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB8	OTFDEC_ R4ENDADDR	REG4_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC	OTFDEC_ R4NONCER0	REG4_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC0	OTFDEC_ R4NONCER1	REG4_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC4	OTFDEC_ R4KEYR0	REG4_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC8	OTFDEC_ R4KEYR1	REG4_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC	OTFDEC_ R4KEYR2	REG4_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD0	OTFDEC_ R4KEYR3	REG4_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4- 0x2FC	Reserved	Reserved																															

Table 409. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	OTFDEC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x304	OTFDEC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x308	OTFDEC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	XONEIE	SEIE
	Reset value																														0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

40 High-resolution timer (HRTIM)

40.1 Introduction

The high-resolution timer can generate up to 12 digital signals with highly accurate timings. It is primarily intended to drive power conversion systems such as switch mode power supplies or lighting systems, but can be of general purpose usage, whenever a very fine timing resolution is expected.

Its modular architecture is used to generate either independent or coupled waveforms. The wave-shape is defined by self-contained timings (using counters and compare units) and a broad range of external events, such as analog or digital feedback and synchronization signals. This is used to produce a large variety of control signals (PWM, phase-shifted, constant Ton,...) and address most of the conversion topologies.

For control and monitoring purposes, the timer also has timing measure capabilities and links to built-in ADC and DAC converters. Last, it features a light-load management mode and is able to handle various fault schemes for safe shut-down purposes.

40.2 Main features

- High-resolution timing units
  - 112 ps resolution, compensated against voltage and temperature variations
  - Duty cycle, frequency, phase shift, and pulse width (triggered one-pulse mode) can be adjusted with full resolution, on all outputs
  - 6 16-bit timing units (each one with an independent counter and four compare units)
  - 12 outputs that can be controlled by any timing unit, up to 32 set/reset sources per channel
  - Modular architecture to address either multiple independent converters with one or two switches or few large multiswitch topologies
- Up to 16 external events, available for any timing unit
  - Suitable for peak current control, zero voltage, and zero current detection (ZVS and ZCS operation) and counter reset (variable frequency operation)
  - 4 sources per external events (digital inputs, built-in comparators outputs, ADC's analog watchdogs, and general purpose timer's trigger outputs)
  - Programmable polarity and edge sensitivity
  - 8 events with a fast asynchronous mode
  - 8 events with a programmable digital filter
  - Spurious events filtering with blanking and windowing modes
- Direct connections to built-in analog peripherals
  - 10 triggers to ADC converters
  - 9 triggers for DAC converters, among which 6 with slope compensation capabilities
  - Comparator outputs (including filtering and blanking features)
- Versatile protection scheme
  - Suitable for overcurrent, short-circuit, and over-voltage/under-voltage protection
  - 8 fault inputs can be combined and associated with any timing unit
  - 3 sources per fault channel (digital inputs, built-in comparators outputs, and external events)
  - Programmable polarity, edge sensitivity, and programmable digital filter
  - Dedicated delayed protections for resonant converters
- Multiple HRTIM instances can be synchronized with external synchronization inputs/outputs
- Versatile output stage
  - High-resolution deadtime insertion (down to 446 ps)
  - Programmable output polarity
  - Chopper mode
- Burst mode controller to handle light-load operation synchronously on multiple converters
- 8 interrupt vectors, each one with up to 14 sources
- 7 DMA requests with up to 14 sources, with a burst mode for multiple registers update

40.3 Functional description

40.3.1 General description

The HRTIM can be partitioned into several subentities:

- The master timer
- The timing units (timer A to timer F)
- The output stage
- The burst mode controller
- An external event and fault signal conditioning logic that is shared by all timers
- The system interface

The master timer is based on a 16-bit up counter. It can set/reset any of the 12 outputs via four compare units and it provides synchronization signals to the six timer units. Its main purpose is to have the timer units controlled by a unique source. An interleaved buck converter is a typical application example where the master timer manages the phase-shifts between the multiple units.

The timer units are working either independently or coupled with the other timers including the master timer. Each timer contains the controls for two outputs. The outputs set/reset events are triggered either by the timing units compare registers or by events coming from the master timer, from the other timers, or from external events.

The output stage has several duties

- Addition of deadtime when the two outputs are configured in complementary PWM mode
- Addition of a carrier frequency on top of the modulating signal
- Management of fault events, by asynchronously asserting the outputs to a predefined safe level

The burst mode controller can take over the control of one or multiple timers in case of light-load operation. The burst length and period can be programmed, as well as the idle state of the outputs.

The external event and fault signal conditioning logic includes:

- The input selection MUXes (for instance for selecting a digital input or an on-chip source for a given external event channel)
- Polarity and edge-sensitivity programming
- Digital filtering (for 5 channels out of 12)

The system interface allows the HRTIM to interact with the rest of the MCU:

- Interrupt requests to the CPU
- DMA controller for automatic accesses to/from the memories, including an HRTIM specific burst mode
- Triggers for the ADC and DAC converters

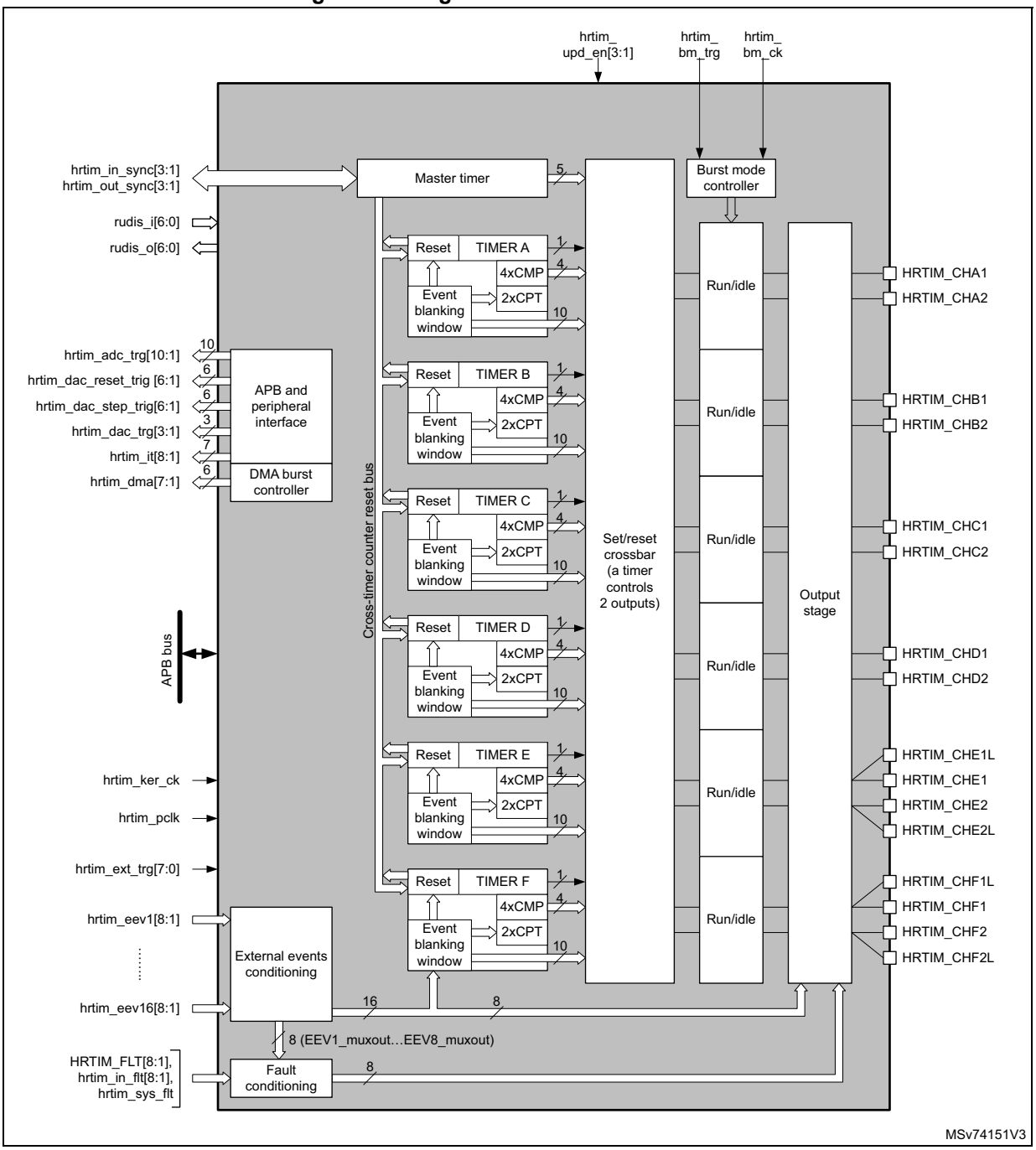
The HRTIM registers are split into eight groups:

- Master timer registers
- Timer A to timer F registers
- Common registers for features shared by all timer units

Note: As a writing convention, references to the six timing units in the text and in the registers are generalized using the "x" letter, where x can be any value from A to F.

The block diagram of the timer is shown in *Figure 354*.

Figure 354. High-resolution timer overview



40.3.2 HRTIM pins and internal signals

The tables in this section summarize the HRTIM inputs and outputs, both on-chip and off-chip.

Table 410. HRTIM inputs/outputs summary

Signal name	Signal type	Description
HRTIM_CHA1, HRTIM_CHA2, HRTIM_CHB1, HRTIM_CHB2, HRTIM_CHC1, HRTIM_CHC2, HRTIM_CHD1, HRTIM_CHD2, HRTIM_CHE1, HRTIM_CHE2, HRTIM_CHF1, HRTIM_CHF2	Outputs	Main HRTIM timer outputs. They can be coupled by pairs (HRTIM_CHx1 & HRTIM_CHx2) with deadtime insertion or work independently.
HRTIM_CHE1L HRTIM_CHE2L HRTIM_CHF1L HRTIM_CHF2L	Outputs	Low-resolution HRTIM timer outputs.These outputs are derived from the main HRTIM outputs before entering the high-resolution stage. The maximum resolution is determined by f_{HRTIM} instead of f_{HRCK}. For more information, see *Section : Timer clock and prescaler*. This section applies only to STM32G53x and STM32G54x devices.
HRTIM_FLT[8:1] hrtim_in_flt[8:1]	Input	Fault inputs: immediately disable the HRTIM outputs when asserted (8 on-chip inputs and 8 off-chip HRTIM_FLTx inputs).
hrtim_sys_flt	Input	System fault gathering MCU internal fault events (clock security system, SRAM, D-cache and flash memory double error detection, Cortex-M33 lockup (HardFault), PVD output.
hrtim_in_sync[3:1]	Input	Synchronization inputs to synchronize the whole HRTIM with other internal or external timer resources: hrtim_in_sync1: Reserved hrtim_in_sync2: the source is an internal timer output hrtim_in_sync3: the source is HRTIM_SCIN input pins or a PLAY output(1)
hrtim_out_sync[2:1]	Output	The purpose of this output is to cascade or synchronize several HRTIM instances, either on-chip or off-chip: hrtim_out_sync1: Reserved hrtim_out_sync2: the destination is an off-chip HRTIM or peripheral (via HRTIM_SCOUT output pins)

Table 410. HRTIM inputs/outputs summary (continued)

Signal name	Signal type	Description
hrtim_eev1[8:1]	Input	External events. Each of the 16 events can be selected among 8 sources, either on-chip (from other built-in peripherals: comparator, ADC analog watchdog, TIMx timers, trigger outputs) or off-chip (HRTIM_EEVx input pins).
hrtim_eev2[8:1]		
hrtim_eev3[8:1]		
hrtim_eev4[8:1]		
hrtim_eev5[8:1]		
hrtim_eev6[8:1]		
hrtim_eev7[8:1]		
hrtim_eev8[8:1]		
hrtim_eev9[8:1]		
hrtim_eev10[8:1]		
hrtim_eev11[8:1]		
hrtim_eev12[8:1]		
hrtim_eev13[8:1]		
hrtim_eev14[8:1]		
hrtim_eev15[8:1]		
hrtim_eev16[8:1]		
hrtim_ext_trg[7:0]	Input	External triggers, for crossbar override and extra faults inputs
hrtim_upd_en[3:1]	Input	A pulse on the update enable inputs hrtim_upd_end[3:1] (on-chip interconnect) triggers the transfer from shadow to active registers
hrtim_bm_trg	Input	A pulse on this input triggers a burst mode entry.
hrtim_bm_ck[4:1]	Input	Burst mode clock (on-chip interconnect)
hrtim_adc_trg[10:1]	Output	ADC start of conversion triggers. The hrtim_adc_trg1 .. hrtim_adc_trg10 outputs are here-after referred to as ADC trigger 1 to ADC trigger 10.
hrtim_dac_trg[3:1]	Output	DAC conversion update triggers
hrtim_dac_reset_trg[6:1] hrtim_dac_step_trg[6:1]	Output	Dual channel DAC triggers
hrtim_it[8:1]	Output	Interrupt requests
hrtim_dma[7:1]	Output	DMA requests
hrtim_pclk	Input	APB clock
hrtim_ker_ck	-	HRTIM kernel clock

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 411. External events mapping and associated features

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F	Source (EExSRC[2:0])							
					Src1(000)	Src2(001)	Src3(010)	Src4(011)	Src5(100)	Src6(101)	Src7(110)	Src8(111)
hrtim_eev1[8:1]	Yes	-	-	-	PB14 PC10 PF3	comp2_out	tim1_trgo	adc1_awd1	play1_out0(1)	TIM1_ BKIN2	-	-
hrtim_eev2[8:1]	Yes	-	-	-	PB15 PF4 PG4	comp4_out	tim2_trgo	adc1_awd2	play1_out1(1)	TIM8_ BKIN2	-	-
hrtim_eev3[8:1]	Yes	-	-	-	PA2 PD3 PE14	comp6_out	tim3_trgo(1)	adc1_awd3	play1_out2(1)	TIM20_ BKIN2(1)	-	-
hrtim_eev4[8:1]	Yes	-	-	-	PA8 PD8 PE7 PF0	comp1_out	comp5_out	adc2_awd1	play1_out3(1)	-	-	-
hrtim_eev5[8:1]	Yes	-	-	-	PB9 PC1 PD0	comp3_out	comp7_out	adc2_awd2	play1_out4(1)	-	-	-
hrtim_eev6[8:1]	-	Yes	Yes	-	PA15 PB1 PD9	comp2_out	comp1_out	adc2_awd3	play1_out5(1)	TIM1_ BKIN1	-	-
hrtim_eev7[8:1]	-	Yes	Yes	-	PA12 PG5 PH0	comp4_out	comp7_out	adc3_awd1(1)	play1_out6(1)	TIM8_ BKIN1	-	-
hrtim_eev8[8:1]	-	Yes	-	Yes	PC0 PD1 PE3	comp6_out	comp3_out	adc3_awd2(1)	play1_out7(1)	TIM20_ BKIN1(1)	-	-

Table 411. External events mapping and associated features (continued)

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F	Source (EExSRC[2:0])							
					Src1(000)	Src2(001)	Src3(010)	Src4(011)	Src5(100)	Src6(101)	Src7(110)	Src8(111)
hrtim_eev9[8:1]	-	Yes	-	Yes	PA9 PC3 PD10	comp5_out	comp8_out	comp4_out	play1_out8(1)	-	-	-
hrtim_eev10[8:1]	-	Yes	-	-	PA7 PC5 PC6 PD5 PE8	comp7_out	tim6_trgo	adc3_awd3(1)	play1_out9(1)	-	-	-
hrtim_eev11[8:1]	Yes	-	-	-	PB3 PD14 PH1	comp7_out	tim9_trgo	adc2_awd1	play1_out10(1)	-	-	-
hrtim_eev12[8:1]	Yes	-	-	-	PD2 PE2 PF11	comp8_out	tim12_trgo	adc2_awd2	play1_out11(1)	-	-	-
hrtim_eev13[4:1]	Yes	-	-	-	PF12	comp5_out	tim15_trgo	adc2_awd3	play1_out12(1)	-	-	-
hrtim_eev14[8:1]	-	Yes	-	-	PC13 PF14	comp8_out	tim1_trgo	adc1_awd1	play1_out13(1)	-	-	-
hrtim_eev15[8:1]	-	Yes	-	-	PB4 PC8 PF13	comp1_out	tim8_trgo	adc1_awd2	play1_out14(1)	-	-	-
hrtim_eev16[4:1]	-	Yes	-	-	PC9 PF14	comp3_out	tim20_trgo(1)	adc1_awd3	play1_out15(1)	-	-	-

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability.

Table 412. Update enable inputs and sources

hrtim_upd_en[3:1]	Update source
hrtim_upd_en1	tim10_oc1 / play1_out14(1)
hrtim_upd_en2	tim11_oc1
hrtim_upd_en3	tim6_trgo

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability. The selection is controlled by the UPDENSEL bit in the *HRTIM option register (HRTIM_OR)*.

Table 413. Burst mode clock sources

hrtim_bm_ck[4:1]	BMCLK[3:0]	Clock source
hrtim_bm_ck1	0110	tim12_oc1
hrtim_bm_ck2	0111	tim13_oc1
hrtim_bm_ck3	1000	tim7_trgo
hrtim_bm_ck4	1001	tim27_trgo

Table 414. Burst mode trigger source

-	trigger source
hrtim_bm_trg	tim26_trgo /play1_out13(1)

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability. The selection is controlled by the BMTRGSEL bit in the *HRTIM option register (HRTIM_OR)*.

Table 415. Internal synchronization source

-	synchornization source
hrtim_in_sync2	tim1_trgo
hrtim_in_sync3	PA15 / PB2 / PF0 / play1_out15(1)

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability. The selection is controlled by the INSYNCSEL bit in the *HRTIM option register (HRTIM_OR)*.

Table 416. Fault inputs

Fault channel	External Input FLTxSRC[1:0] = 00 (HRTIM_FLT[8:1])	On-chip source FLTxSRC[1:0] = 01 (hrtim_flt_in[8:1])	External Input FLTxSRC[1:0] = 10 (HRTIM internal(1))	On-chip source FLTxSRC[1:0] = 11 hrtim_ext_trg[7:0]
FAULT 1	PA11 PC4 PD13 PG9 PH1	comp2_out	EEV1_mux_out	hrtim_ext_trg0
FAULT 2	PB12 PE15	comp4_out	EEV2_mux_out	hrtim_ext_trg1
FAULT 3	PA9 PB10 PD4 PE9	comp6_out	EEV3_mux_out	hrtim_ext_trg2
FAULT 4	PB11 PD15 PG14	comp1_out	EEV4_mux_out	hrtim_ext_trg3
FAULT 5	PB0 PB3 PC7 PD6	comp3_out	EEV5_mux_out	hrtim_ext_trg4
FAULT 6	PB2 PB13 PE13	comp5_out	EEV6_mux_out	hrtim_ext_trg5
FAULT 7	PA13 PC14 PE10	comp7_out	EEV7_mux_out	hrtim_ext_trg6
FAULT 8	PC15 PE11	comp8_out	EEV8_mux_out	hrtim_ext_trg7

1. Refer to *Section 40.3.17: Fault protection* for details.

Table 417. External trigger inputs

External trigger	Source
hrtim_ext_trg0	play1_out0
hrtim_ext_trg1	play1_out1
hrtim_ext_trg2	play1_out2
hrtim_ext_trg3	play1_out3
hrtim_ext_trg4	play1_out4
hrtim_ext_trg5	play1_out5

Table 417. External trigger inputs (continued)

External trigger	Source
hrtim_ext_trg6	play1_out6
hrtim_ext_trg7	play1_out7

Table 418. HRTIM DAC triggers connections

HRTIM DAC triggers	DAC1	DAC2	DAC5	DAC6	DAC7	DAC8	DAC9	DAC10
hrtim_dac_trg1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
hrtim_dac_trig2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
hrtim_dac_trig3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
hrtim_dac_reset_trgA hrtim_dac_step_trgA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
hrtim_dac_reset_trgB hrtim_dac_step_trgB	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
hrtim_dac_reset_trgC hrtim_dac_step_trgC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
hrtim_dac_reset_trgD hrtim_dac_step_trgD	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
hrtim_dac_reset_trgE hrtim_dac_step_trgE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
hrtim_dac_reset_trgF hrtim_dac_step_trgF	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

40.3.3 Clocks

The HRTIM must be supplied by the highest possible kernel clock frequency hrtim_ker_ck (here-after referred as f_{HRTIM}) to offer full resolution. The t_{HRTIM} clock period is evenly divided into up to 32 intermediate steps using an edge positioning logic. All clocks present in the HRTIM are derived from this reference clock.

Definition of terms

f_{HRTIM}: Main HRTIM clock (hrtim_ker_ck). All subsequent clocks are derived and synchronous with this source.

f_{HRCK}: High-resolution equivalent clock. Considering the f_{HRTIM} clock period division by 32, it is equivalent to a frequency of 32 × f_{HRTIM}.

f_{DTG}: Deadtime generator clock. For convenience, only the t_{DTG} period (t_{DTG} = $1/f_{DTG}$) is used in this document.

f_{CHPFRQ}: Chopper stage clock source.

f_{1STPW}: Clock source defining the length of the initial pulse in chopper mode. For convenience, only the t_{1STPW} period (t_{1STPW} = $1/f_{1STPW}$) is used in this document.

f_{BRST}: Burst mode controller counter clock.

$f_{SAMPLING}$: Clock needed to sample the fault or the external events inputs.

f_{FLTS}: Clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter fault events.

f_{EEVS}: Clock derived from f_{HRTIM} which is used as a source for $f_{SAMPLING}$ to filter external events.

Timer clock and prescaler

Each timer in the HRTIM has its own individual clock prescaler, which is used to adjust the timer resolution (see *Table 419*).

Table 419. Timer resolution and min. PWM frequency for f_{HRTIM} = 280 MHz

CKPSC[2:0]	Prescaling ratio	f_{HRCK} equivalent frequency	Resolution	Min PWM frequency
000	1	280 × 32 MHz = 8.96 GHz	112 ps	136 kHz
001	2	280 × 16 MHz = 4.48 GHz	225 ps	68 kHz
010	4	280 × 8 MHz = 2.24 GHz	450 ps	34 kHz
011	8	280 × 4 MHz = 1.12 GHz	900 ps	17 kHz
100	16	280 × 2 MHz = 560 MHz	1.8 ns	8.5 kHz
101	32	280 MHz	3.6 ns	4.2 kHz
110	64	280/2 MHz = 140 MHz	7.2 ns	2.12 kHz
111	128	280/4 MHz = 70 MHz	14.4 ns	1.06 kHz

The high-resolution is available for edge positioning, PWM period adjustment, and externally triggered pulse duration.

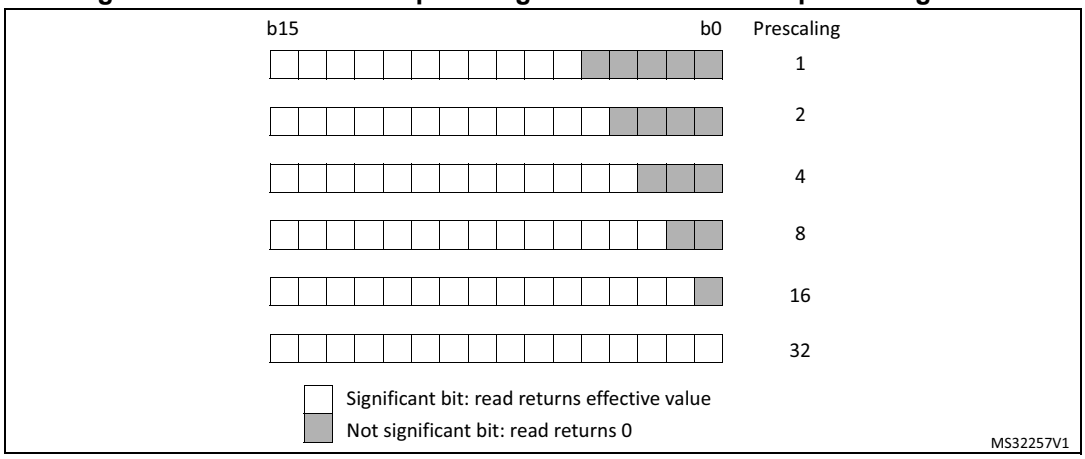
The high-resolution is not available for the following features

- Timer counter read and write accesses
- Capture unit

For clock prescaling ratios below 32 (CKPSC[2:0] < 5), the least significant bits of the counter and capture registers are not significant. The least significant bits cannot be written (counter register only) and return 0 when read.

For instance, if CKPSC[2:0] = 2 (prescaling by 4), writing 0xFFFF into the counter register yields an effective value of 0xFFF8. Conversely, any counter value between 0xFFFF and 0xFFF8 is read as 0xFFF8.

Figure 355. Counter and capture register format vs clock prescaling factor



Low-resolution HRTIM timer outputs

Besides the main outputs, alternate low-resolution outputs are available on some packages. These outputs operate as if the clock prescaler is set to a value greater than or equal to five (CKPSC[2:0] ≥ 5), with the same functionality as the main outputs and a maximum resolution determined by f_{HRTIM} instead of f_{HRCK}.

These outputs are relevant for applications that operate with limited PWM frequencies and do not require high resolution.

Note: It is not mandatory to set CKPSC to a value greater than or equal to five to use these outputs. Setting an identical CKPSC value for all timing units allows events to propagate from one timing unit to another (see warning below).

Initialization

At startup, it is mandatory to initialize first the prescaler bitfields before writing the compare and period registers. Once the timer is enabled (MCEN or TxCEN bit set in the HRTIM_MCR register), the prescaler cannot be modified.

When multiple timers are enabled, the prescalers are synchronized with the prescaler of the timer that was started first.

Warning: It is possible to have different prescaling ratios in the master and TIMA..E timers only if the counter and output behavior does not depend on other timers' information and signals. It is mandatory to configure identical prescaling ratios in these timers when one of the following events is propagated from one timing unit (or master timer) to another: output set/reset event, counter reset event, update event, external event filter, or capture triggers. Prescaler factors not equal yield to unpredictable results.

Deadtime generator clock

The deadtime prescaler is supplied by (f_{HRTIM} × 8) / $2^{(DTPRSC[2:0])}$, programmed with DTPRSC[2:0] bits in the HRTIM_DTxR register.

t_{DTG} ranges from 446 ps to 57 ns for f_{HRTIM} = 280 MHz.

Chopper stage clock

The chopper stage clock source f_{CHPFRQ} is derived from f_{HRTIM} with a division factor ranging from 16 to 256, so that 1.09 MHz ≤ f_{CHPFRQ} ≤ 17.5 MHz for f_{HRTIM} = 280 MHz.

t_{1STPW} is the length of the initial pulse in chopper mode, programmed with the STRPW[3:0] bits in the HRTIM_CHPxR register, as follows:

t_{1STPW} = (STRPW[3:0]+1) × 16 × t_{HRTIM}.

It uses f_{HRTIM} / 16 as clock source (17.5 MHz for f_{HRTIM}= 280 MHz).

Burst mode prescaler

The burst mode controller counter clock f_{BRST} can be supplied by several sources, among which one is derived from f_{HRTIM}.

In this case, f_{BRST} ranges from f_{HRTIM} to f_{HRTIM} / 32768 (8.55 kHz for f_{HRTIM} = 280 MHz).

Fault input sampling clock

The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{FLTS}.

f_{FLTS} is derived from f_{HRTIM} and ranges from 280 MHz to 35 MHz for f_{HRTIM} = 280 MHz.

External event input sampling clock

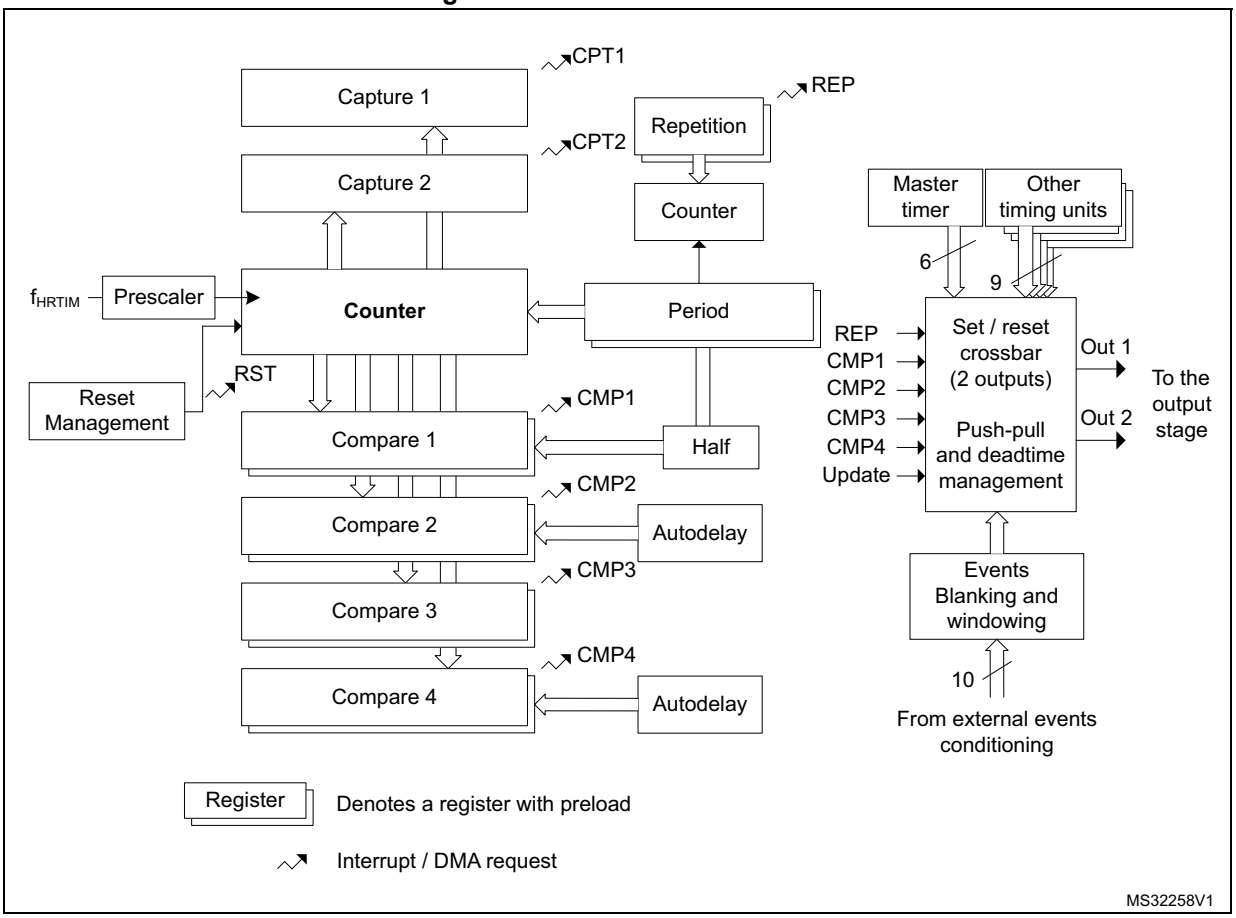
The fault input noise rejection filter has a time constant defined with $f_{SAMPLING}$ which can be either f_{HRTIM} or f_{EEVS}.

f_{EEVS} is derived from f_{HRTIM} and ranges from 280 MHz to 35 MHz for f_{HRTIM} = 280 MHz.

40.3.4 Timer A..F timing units

The HRTIM embeds six identical timing units made of a 16-bit upcounter with an autoreload mechanism to define the counting period, four compare, and two capture units, as per *Figure 356*. Each unit includes all control features for 2 outputs, so that it operates as a standalone timer.

Figure 356. Timer A..F overview



The period and compare values must be within a lower and an upper limit related to the high-resolution implementation and listed in *Table 420*:

- The minimum value must be greater than or equal to three periods of the f_{HRTIM} clock. The value 0x0000 can be written in CMP1 and CMP3 registers only, to skip a PWM pulse. See *Section : Null duty cycle exception case* for details
- The maximum value must be less than or equal to 0xFFFF - 1 periods of the f_{HRTIM} clock.

Table 420. Period and compare registers min and max values

CKPSC[2:0] value	Min(1)	Max
0	0x0060	0xFFDF
1	0x0030	0xFFEF
2	0x0018	0xFFF7
3	0x000C	0xFFFB
4	0x0006	0xFFFD
≥ 5	0x0003	0xFFFD

1. The value 0x0000 can be written in CMP1 and CMP3 registers only, to skip a PWM pulse. See *Section : Null duty cycle exception case* for details.

Note: A compare value greater than the period register value does not generate a compare match event.

Counter operating mode

Timer A..F operate in continuous (free-running) mode or in single-shot manner where counting is started by a reset event, using the CONT bit in the HRTIM_TIMxCR control register. An additional RETRIG bit allows you to select whether the single-shot operation is retriggerable or nonretriggerable. Details of the operation are summarized on *Table 421* and on *Figure 357* and *Figure 358*.

Table 421. Timer operating modes

CONT	RETRIG	Operating mode	Start / stop conditions Clocking and event generation
0	0	Single-shot Non-retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A first reset event starts the counting and any subsequent reset is ignored until the counter reaches the PER value. The PER event is then generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
0	1	Single-shot Retriggerable	Setting the TxEN bit enables the timer but does not start the counter. A reset event starts the counting if the counter is stopped, otherwise it clears the counter. When the counter reaches the PER value, the PER event is generated and the counter is stopped. A reset event re-starts the counting operation from 0x0000.
1	X	Continuous mode	Setting the TxEN bit enables the timer and starts the counter simultaneously. When the counter reaches the PER value, it rolls-over to 0x0000 and resumes counting. The counter can be reset at any time.

The TxEN bit can be cleared at any time to disable the timer and stop the counting.

Figure 357. Continuous timer operation

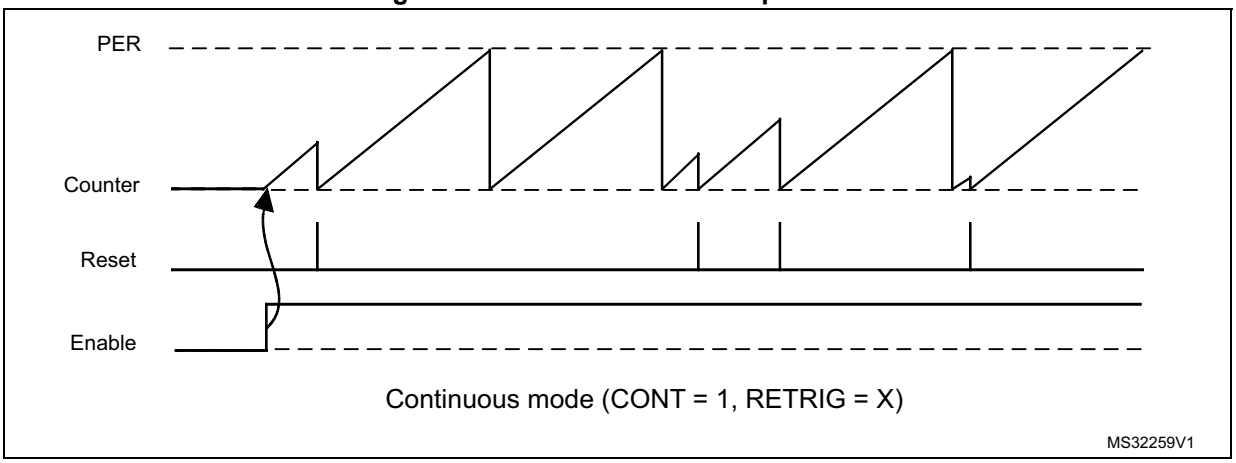
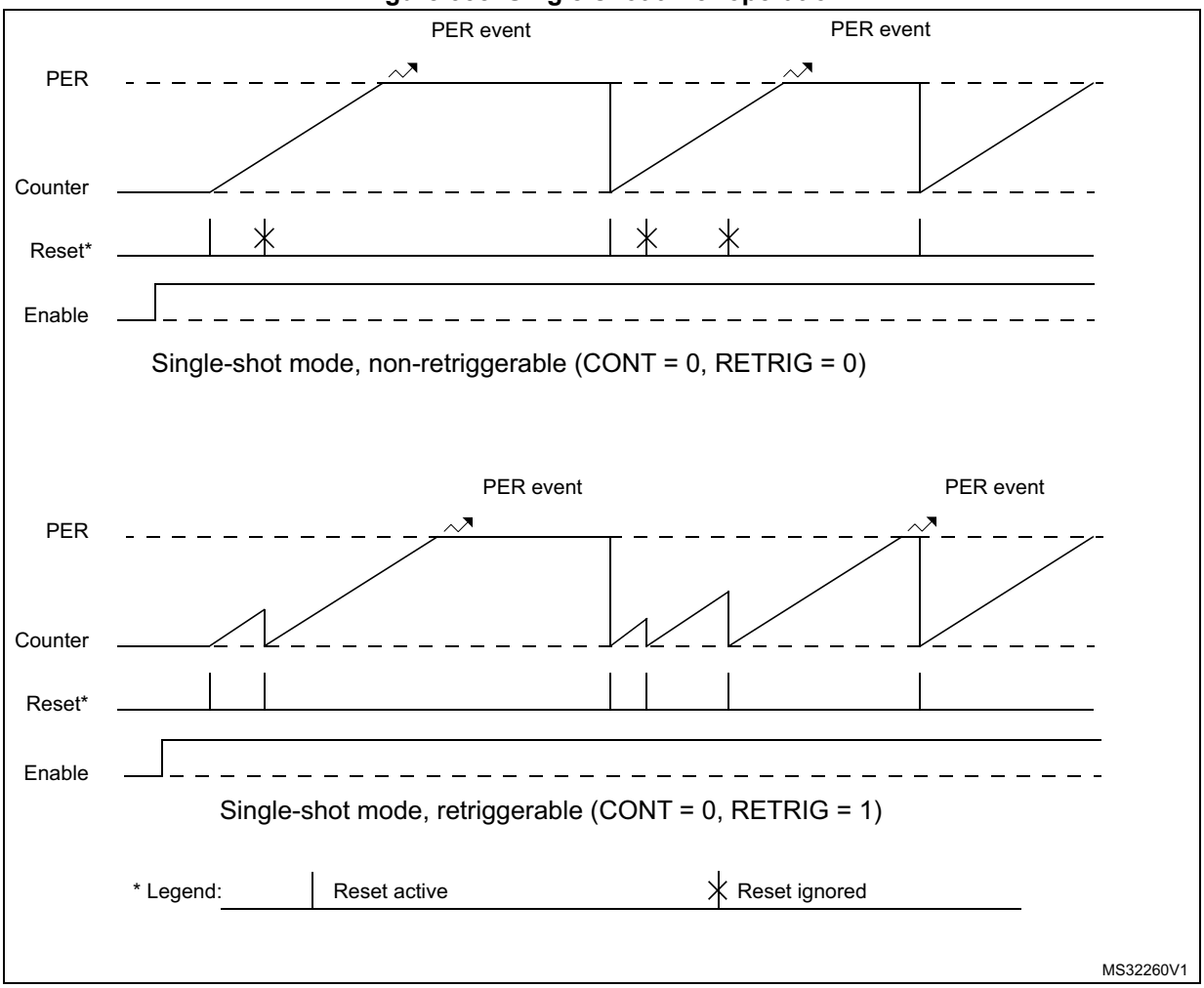


Figure 358. Single-shot timer operation



Roll-over event

A counter roll-over event is generated when the counter goes back to 0 after having reached the period value set in the HRTIM_PERxR register in continuous mode.

This event is used for multiple purposes in the HRTIM:

- To set/reset the outputs
- To trigger the register content update (transfer from preload to active)
- To trigger an IRQ or a DMA request
- To serve as a burst mode clock source or a burst start trigger
- As an ADC trigger
- To decrement the repetition counter

If the initial counter value is above the period value when the timer is started, or if a new period is set while the counter is already above this value, the counter is not reset: it overflows at the maximum period value and the repetition counter does not decrement.

Timer reset

The reset of the timing unit counter can be triggered by up to 30 events that can be selected simultaneously in the HRTIM_RSTxR register, among the following sources:

- The timing unit: compare 2, compare 4 and update (three events)
- The master timer: reset and compare 1..4 (five events)
- The external events EXTEVNTA..J (10 events)
- All other timing units (e.g. timer B..F for timer A): compare 1, 2 and 4 (12 events)

Several events can be selected simultaneously to handle multiple reset sources. In this case, the multiple reset requests are ORed. When two counter reset events are generated within the same f_{HRTIM} clock cycle, the last counter reset is taken into account.

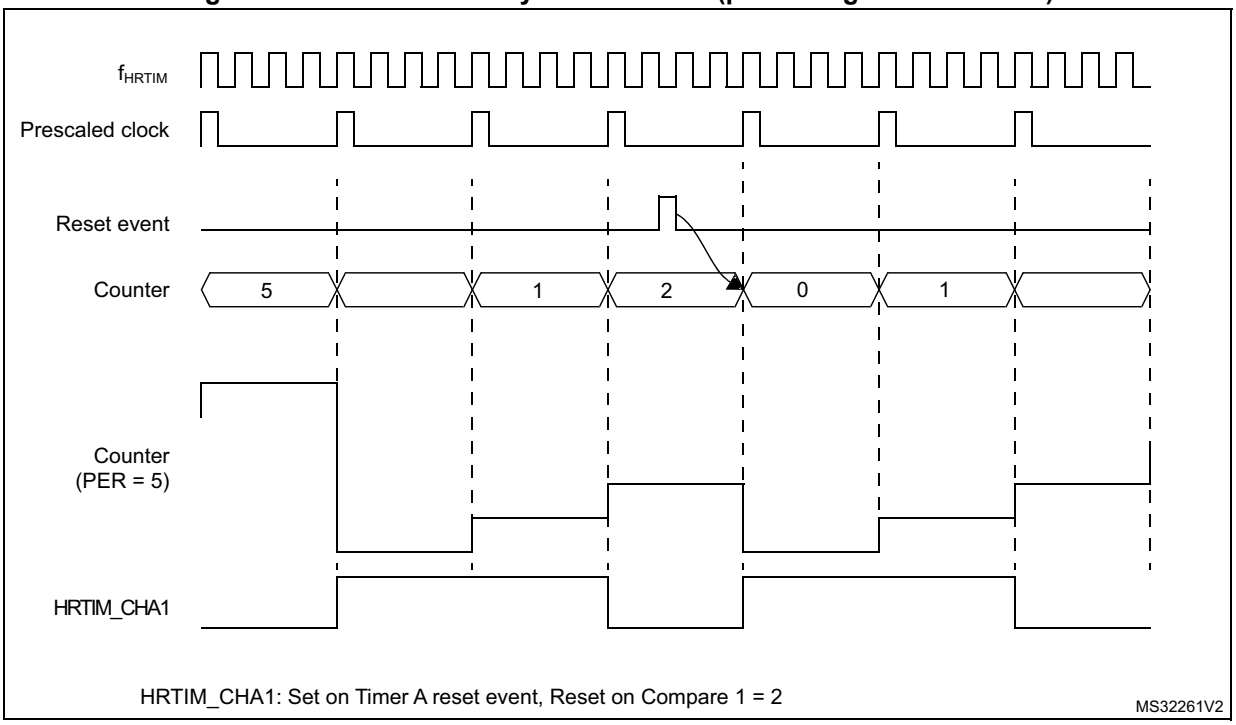
Additionally, it is possible to do a software reset of the counter using the TxRST bits in the HRTIM_CR2 register. These control bits are grouped into a single register to allow the simultaneous reset of several counters.

The reset requests are taken into account only once the related counters are enabled (TxCEN bit set).

When the f_{HRTIM} clock prescaling ratio is above 32 (counting period above f_{HRTIM}), the counter reset event is delayed to the next active edge of the prescaled clock. This is used to maintain a jitterless waveform generation when an output transition is synchronized to the reset event (typically a constant Ton time converter).

Figure 359 shows how the reset is handled for a clock prescaling ratio of 128 (f_{HRTIM} divided by 4).

Figure 359. Timer reset resynchronization (prescaling ratio above 32)



Repetition counter

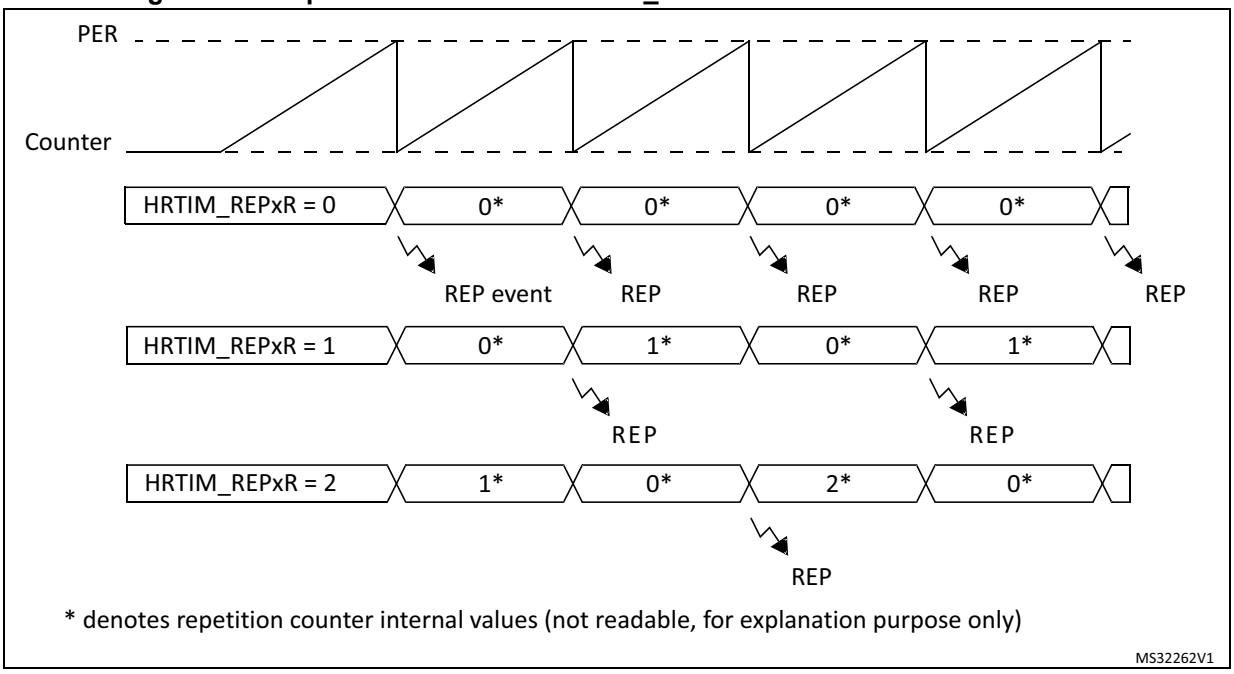
A common software practice is to have an interrupt generated when the period value is reached, so that the maximum amount of time is left for processing before the next period begins. The main purpose of the repetition counter is to adjust the period interrupt rate and off-load the CPU by decoupling the switching frequency and the interrupt frequency.

The timing units have a repetition counter. This counter cannot be read, but solely programmed with an autoreload value in the HRTIM_REPxR register.

The repetition counter is initialized with the content of the HRTIM_REPxR register when the timer is enabled (TXCEN bit set). Once the timer has been enabled, anytime the counter is cleared, either due to a reset event or due to a counter roll-over, the repetition counter is decreased. When it reaches zero, a REP interrupt or a DMA request is issued if enabled (REPIE and REPDE bits in the HRTIM_DIER register).

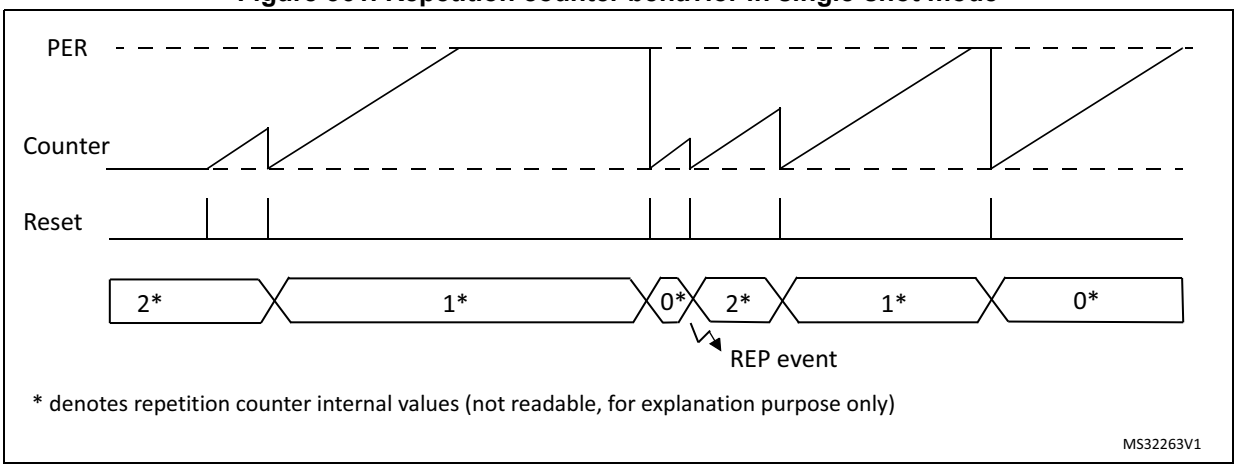
If the HRTIM_REPxR register is set to 0, an interrupt is generated for each and every period. For any value above 0, a REP interrupt is generated after (HRTIM_REPxR + 1) periods. *Figure 360* presents the repetition counter operation for various values, in continuous mode.

Figure 360. Repetition rate versus HRTIM_REPxR content in continuous mode



The repetition counter can also be used when the counter is reset before reaching the period value (variable frequency operation) either in continuous or in single-shot mode (*Figure 361* here-below). The reset causes the repetition counter to be decremented, at the exception of the very first start following counter enable (TxCEN bit set).

Figure 361. Repetition counter behavior in single-shot mode



A reset or start event from the HRTIM_SCIN[3:1] source causes the repetition to be decremented as any other reset. However, in SYNCIN-started single-shot mode (SYNCSTRTx bit set in the HRTIM_TIMxCR register), the repetition counter is decremented only on the first reset event following the period. Any subsequent restarted by a new request on HRTIM_SCIN[3:1] inputs.

Set/reset crossbar

A “set” event correspond to a transition to the output active state, while a “reset” event corresponds to a transition to the output inactive state.

The polarity of the waveform is defined in the output stage to accommodate positive or negative logic external components: an active level corresponds to a logic level 1 for a positive polarity (POLx = 0), and to a logic level 0 for a negative polarity (POLx = 1).

Each of the timing units handles the set/reset crossbar for two outputs. These two outputs can be set, reset, or toggled by up to 32 events that can be selected among the following sources:

- The timing unit: period, compare 1..4, register update (six events)
- The master timer: period, compare 1..4, HRTIM synchronization (six events)
- All other timing units (e.g. timer B..F for timer A): TIMEVNT1..9 (nine events described in *Table 422*)
- The external events EXTEVNTA..J (10 events)
- A software forcing (one event)

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

The event sources are ORed and multiple events can be simultaneously selected.

Each output is controlled by two 32-bit registers, one coding for the set (HRTIM_SETxyR) and another one for the reset (HRTIM_RSTxyR), where x stands for the timing unit: A..F and y stands for the output 1or 2 (e.g. HRTIM_SETA1R, HRTIM_RSTC2R,...).

If the same event is selected for both set and reset, it toggles the output. It is not possible to toggle the output state more than one time per t_{HRTIM} period: in case of two consecutive toggling events within the same cycle, only the first one is considered.

The set and reset requests are taken into account only once the counter is enabled (TxCEN bit set), except if the software is forcing a request to allow the prepositioning of the outputs at timer startup.

Table 422 summarizes the events from other timing units that can be used to set and reset the outputs. The number corresponds to the timer events (such as TIMEVNTx) listed in the register, and empty locations are indicating nonavailable events.

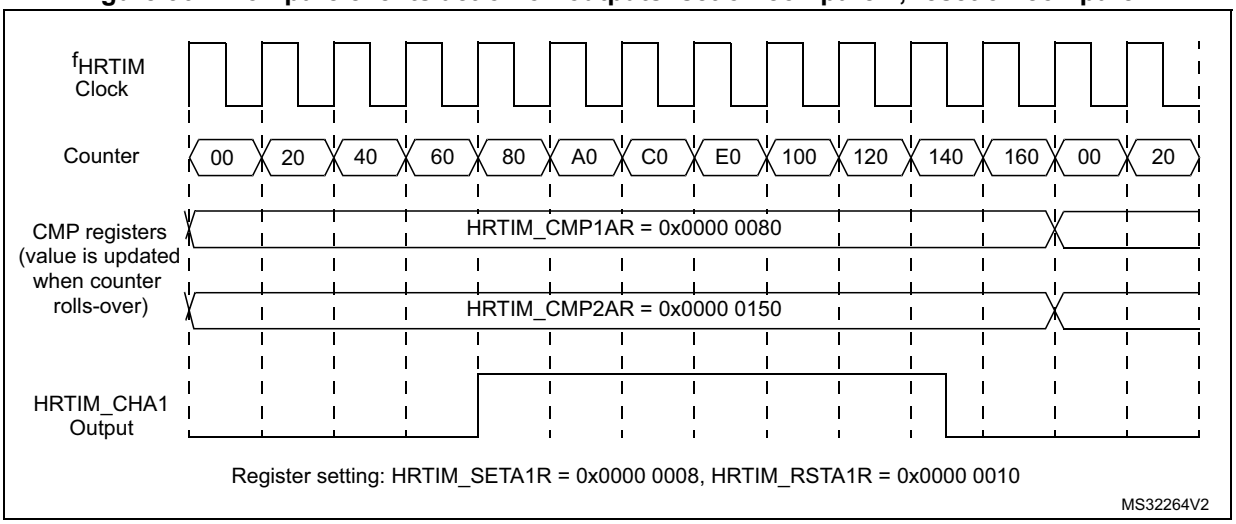
For instance, timer A outputs can be set or reset by the following events: timer B compare 1 and 2, timer C compare 2 and 3,... and timer E compare 3 is listed as TIMEVNT7 in HRTIM_SETA1R.

Table 422. Events mapping across timer A to F

Source		Timer A				Timer B				Timer C				Timer D				Timer E				Timer F			
		CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4	CMP1	CMP2	CMP3	CMP4
Destination	TA	-	-	-	-	1	2	-	-	-	3	4	-	5	6	-	-	-	-	7	8	-	-	-	9
	TB	1	2	-	-	-	-	-	-	-	-	3	4	-	-	5	6	7	8	-	-	-	-	9	-
	TC	-	1	2	-	-	3	4	-	-	-	-	-	-	5	-	6	-	-	7	8	-	9	-	-
	TD	1	-	-	2	-	3	-	4	-	-	-	5	-	-	-	-	6	-	-	7	8	-	9	-
	TE	-	-	-	1	-	-	2	3	4	5	-	-	6	7	-	-	-	-	-	-	-	-	8	9
	TF	-	-	1	-	2	-	-	3	4	-	-	5	-	-	6	7	-	8	9	-	-	-	-	-

Figure 362 represents how a PWM signal is generated using two compare events.

Figure 362. Compare events action on outputs: set on compare 1, reset on compare 2



Set/reset on update events

A set or reset event on update is done at low resolution. When CKPSC[2:0] < 5, the high-resolution delay is set to its maximum value so that a set/reset event on update always lags as compared to other compare set/reset events, with a jitter varying between 0 and 31/32 of a f_{HRTIM} clock period.

Half mode

This mode aims at generating a square signal with a fixed 50% duty cycle and variable frequency (typically for converters using resonant topologies). It is used to have the duty cycle automatically forced to half of the period value when a new period is programmed.

This mode is enabled by writing HALF bit to 1 in the HRTIM_TIMxCR register. When the HRTIM_PERxR register is written, it causes an automatic update of the compare one value with HRTIM_PERxR/2 value.

The output on which a square wave is generated must be programmed to have one transition on the CMP1 event, and one transition on the period event, as follows:

- HRTIM_SETxyR = 0x0000 0008, HRTIM_RSTxyR = 0x0000 0004, or
- HRTIM_SETxyR = 0x0000 0004, HRTIM_RSTxyR = 0x0000 0008

The HALF mode overrides the content of the HRTIM_CMP1xR register. The access to the HRTIM_PERxR register only causes compare 1 internal register to be updated. The user-accessible HRTIM_CMP1xR register is not updated with the HRTIM_PERxR / 2 value.

When the preload is enabled (PREEN = 1), compare 1 active register is refreshed on the update event. If the preload is disabled (PREEN= 0), compare 1 active register is updated as soon as HRTIM_PERxR is written.

The period must be greater than or equal to six periods of the f_{HRTIM} clock (0xC0 if CKPSC[2:0] = 0, 0x60 if CKPSC[2:0] = 1, 0x30 if CKPSC[2:0] = 2,...) when the HALF mode is enabled.

Interleaved mode

This mode complements the Half mode and helps the implementation of interleaved topologies.

It is used to recompute automatically the content of compare registers when the HRTIM_PERxR value is updated.

The selection is done using the HALF bit and the INTLVD[1:0] bits in HRTIM_MCR and HRTIM_TIMxCR registers, as shown on the *Table 423* below.

Table 423. Interleaved mode selection

HALF bit	INTLVD [1:0] bits	Mode
0	00	Disabled
0	01	Triple interleaved (120°)
0	10	Quad interleaved (90°)
0	11	Reserved
1	xx	Dual interleaved (180°)

Table 424 gives the compare values for the three available modes. The content of the compare registers is overridden. The corresponding compare events can be used to trigger an output set/reset or to reset a slave timer.

Table 424. Compare 1..3 values in interleaved mode

Mode	Dual interleaved 180°	Triple interleaved 120°	Quad interleaved 90°
CMP1 value	PERxR/2	PERxR/3	PERxR/4
CMP2 value	Not affected	2x (PERxR/3)	PERxR/2
CMP3 value	Not affected	Not affected	3x (PERxR/4)

Note: In half and interleaved modes, the compare registers are controlled by hardware and writing them has no effect. However the written value is stored in the preload register and becomes active on the update event following the exit of these modes.

Note: The triple and quad interleaved modes must not be used simultaneously with other modes using CMP2 (dual channel dac trigger and triggered-half modes).

Null duty cycle exception case

The high-resolution behavior is not supported for pulses narrower than three t_{HRTIM} periods (see *Section 40.3.7: Set/reset events priorities and narrow pulses management*) and any value strictly below 3 periods of the f_{HRTIM} clock (that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...) in the HRTIM_TIMxCMPy register is forbidden (see *40.5.19: HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)*).

However, it is possible to skip an output pulse and have a null duty cycle by simply writing a null value in the HRTIM_TIMxCMPy registers, if and only if the following conditions are met:

- The output SET event is generated by the PERIOD event
- The output RESET if generated by the compare event
- The compare event is active within the timer unit itself, and not used for other timing units

For any other use case, this can be done by programming the SET and RESET events with the very same compare values, above three periods of the f_{HRTIM} clock. In this case, the output is reset forced (following the set/reset priority scheme defined in the *Section 40.3.7: Set/reset events priorities and narrow pulses management*).

When a null value is programmed, the CMPx flags are set, and the related interrupt is issued if enabled. An event is also generated on synchronization output when SYNCSRC= 11 (synchronization generated on TIM1 CMP1 event).

The following limitations apply when a null value is programmed:

- ADC and burst mode triggers are not generated
- The CMP1 cannot be used for blanking events or faults

Swap mode

This mode is used to swap the two outputs with a single bit access: the output 1 signal is connected to the output 2 pin and the output 2 signal is connected to output 1 pin. The output swap is triggered with the SWPx bits in the HRTIM_CR2 register and is effective on the next update event.

The outputs are swapped prior to the set/reset crossbar unit, as follows:

- If SWPx = 0, HRTIM_SETx1R and HRTIM_RSTx1R are coding for the output 1, HRTIM_SETx2R and HRTIM_RSTx2R are coding for the output 2
- If SWPx = 1, HRTIM_SETx1R and HRTIM_RSTx1R are coding for the output 2, HRTIM_SETx2R and HRTIM_RSTx2R are coding for the output 1

The swap mode is only affecting the preload register, and not the active registers.

Note: The preload mode must be enabled when using the swap mode.

Consequently, it does not modify the auxiliary outputs in parallel with the regular outputs going to the output stage (see *Section 40.3.18* for details). They provide the following internal status, events, and signals:

- O1CPY, O2CPY, SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset (TA2, TB2, TC2, TD2, TE2, TF2)
- External event filters generated with a Tx2 output copy

For instance the SETx1 flag is related to the output 1 when SWP = 0 and is related to the output 2 when SWPx = 1.

Similarly, the swap mode does not change the attribution of control bits in the HRTIM_OUTxR register (DIDLx, CHPx, FAULTx[1:0], IDLESx, POLx bits). For instance, the POL1 bit controls the output 1 polarity whatever the SWP bit value.

Note: The SWPx bits are ignored in push-pull mode (PSHPLL = 1 in the HRTIM_TIMxCR register).

Capture

The timing unit has the capability to capture the counter value, triggered by internal and external events. The purpose is to:

- Measure events arrival timings or occurrence intervals
- Update compare 2 and compare 4 values in autodelayed mode (see *Section : Autodelayed mode*).

The capture is done with f_{HRTIM} resolution: for a clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the register are not significant (read as 0).

The timer has two capture registers: HRTIM_CPT1xR and HRTIM_CPT2xR. The capture triggers are programmed in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers.

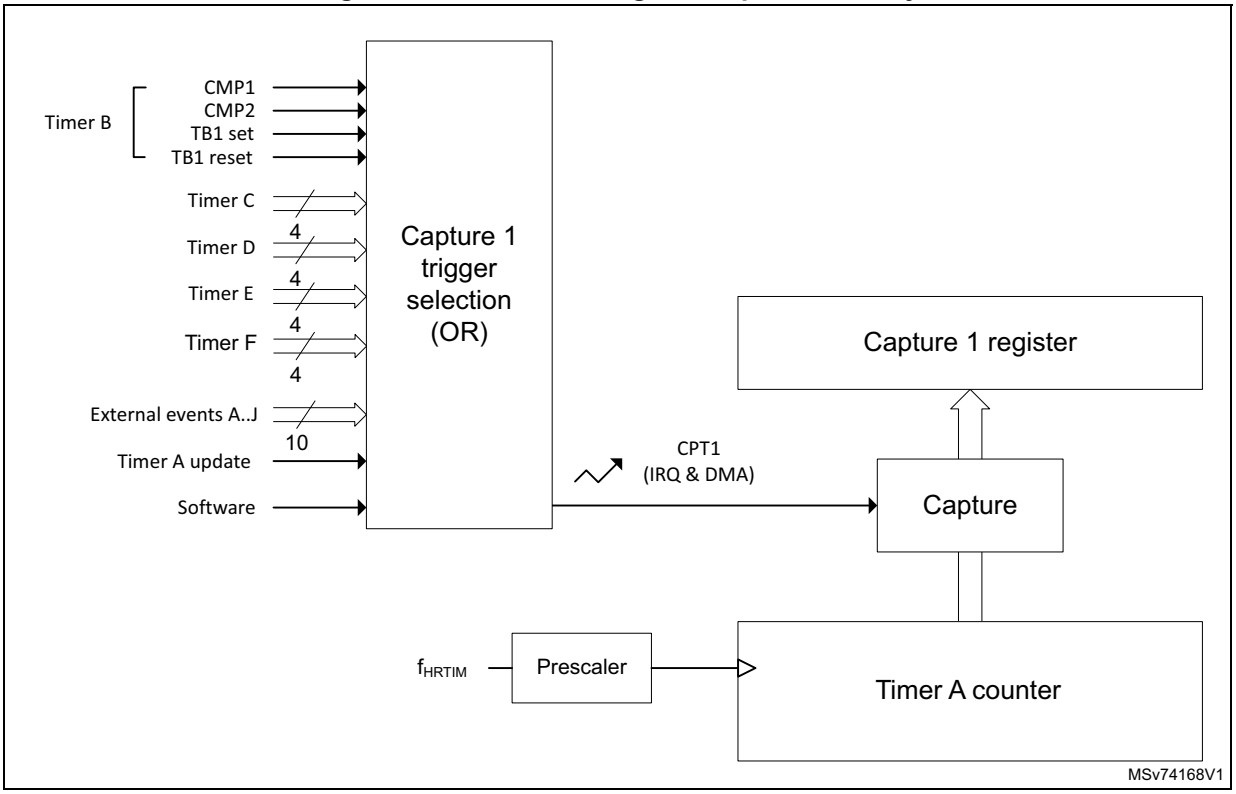
The capture of the timing unit counter can be triggered by up to 28 events that can be selected simultaneously in the HRTIM_CPT1xCR and HRTIM_CPT2xCR registers, among the following sources:

- The external events, EXTEVNTA..J (10 events)
- All other timing units (e.g. timer B..F for timer A): compare 1, 2 and output 1 set/reset events (16 events)
- The timing unit: update (one event)
- A software capture (one event)

Several events can be selected simultaneously to handle multiple capture triggers. In this case, the concurrent trigger requests are ORed. The capture can generate an interrupt or a DMA request when CPTxIE and CPTxDE bits are set in the HRTIM_TIMxDIER register.

Overcapture is not prevented by the circuitry: a new capture is triggered even if the previous value was not read, or if the capture flag was not cleared.

Figure 363. Timer A timing unit capture circuitry



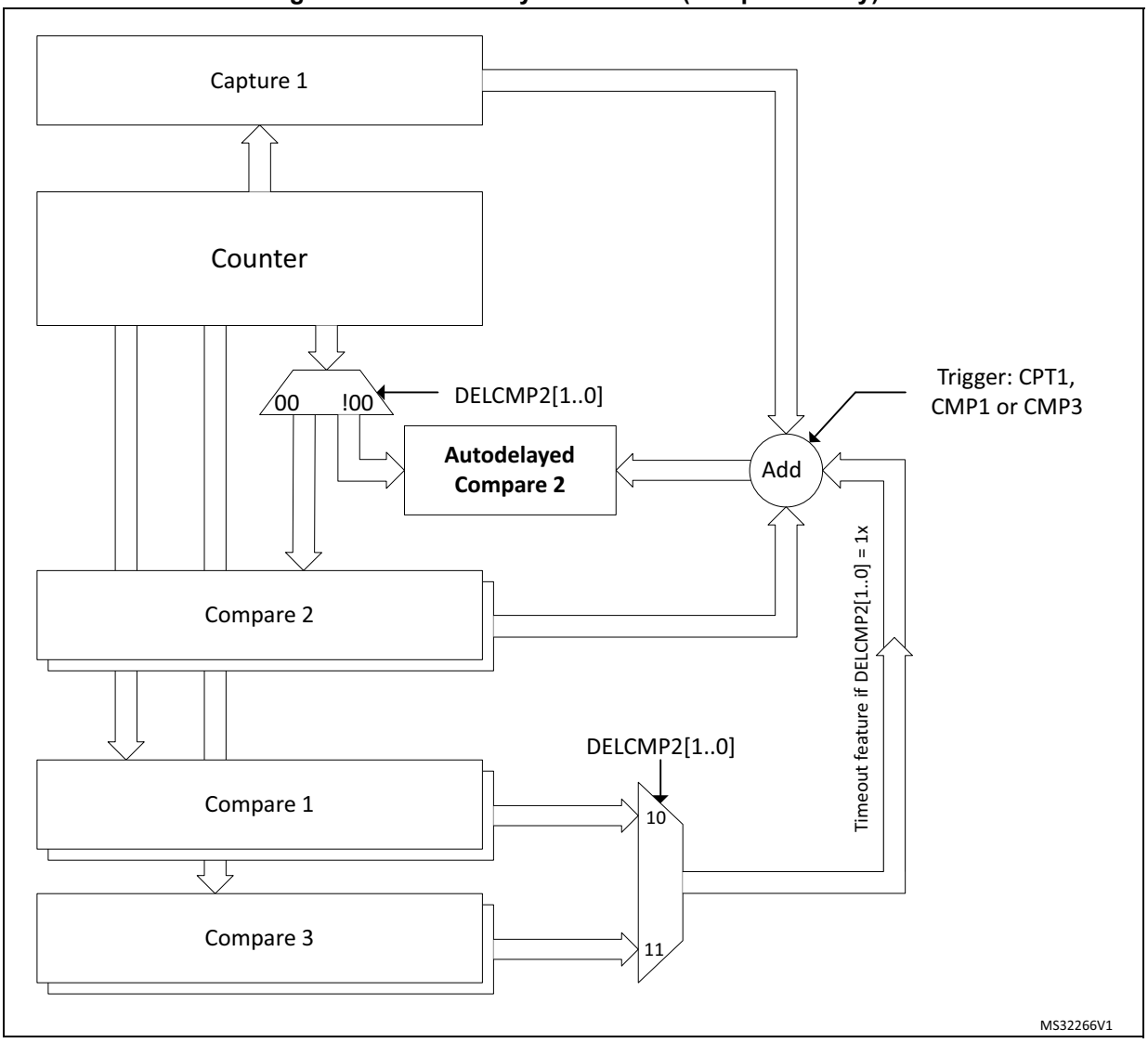
Autodelayed mode

This mode is used to have compare events generated relatively to capture events, so that for instance an output change can happen with a programmed timing following a capture. In this case, the compare match occurs independently from the timer counter value. It enables the generation of waveforms with timings synchronized to external events without the need of software computation and interrupt servicing.

As long as no capture is triggered, the content of the HRTIM_CMPxR register is ignored (no compare event is generated when the counter value matches the compare value. Once the capture is triggered, the compare value programmed in HRTIM_CMPxR is summed with the captured counter value in HRTIM_CPTxyR, and it updates the internal autodelayed compare register, as seen on *Figure 364*. The autodelayed compare register is internal to the timing unit and cannot be read. The HRTIM_CMPxR preload register is not modified after the calculation.

This feature is available only for compare 2 and compare 4 registers. Compare 2 is associated with capture 1, while compare 4 is associated with capture 2. HRTIM_CMP2xR and HRTIM_CMP4xR compares cannot be programmed with a value below three f_{HRTIM} clock periods, as in the regular mode.

Figure 364. Auto-delayed overview (compare 2 only)



The autodelayed compare is only valid from the capture up to the period event: once the counter has reached the period value, the system is rearmed with compare disabled until a capture occurs.

DELCMP2[1:0] and DELCMP4[1:0] bits in HRTIM_TIMxCR register are used to configure the autodelayed mode as follows:

- 00
  Regular compare mode: HRTIM_CMP2xR and HRTIM_CMP4xR register contents are directly compared with the counter value.
- 01
  Autodelayed mode: compare 2 and compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event.
- 10 or 11
  Autodelayed mode with timeout: compare 2 and compare 4 values are recomputed and used for comparison with the counter after a capture 1/2 event or after a compare 1

match (DELCMPx[1:0]= 10) or a compare 3 match (DELCMPx[1:0]= 11) to have a timeout function if capture 1/2 event is missing.

When the capture occurs, the comparison is done with the (HRTIM_CMP2/4xR + HRTIM_CPT1/2xR) value. If no capture is triggered within the period, the behavior depends on the DELCMPx[1:0] value:

- DELCMPx[1:0] = 01: the compare event is not generated
- DELCMPx[1:0] = 10 or 11: the comparison is done with the sum of the two compares (for instance HRTIM_CMP2xR + HRTIM_CMP1xR). The captures are not taken into account if they are triggered after CMPx + CMP1 (resp. CMPx + CMP3).

The captures are enabled again at the beginning of the next PWM period.

If the result of the autodelayed summation is above 0xFFFF (overflow), the value is ignored and no compare event is generated until a new period is started.

Note: DELCMPx[1:0] bitfield must be reset when reprogrammed from one value to the other to reinitialize properly the autodelayed mechanism, for instance:

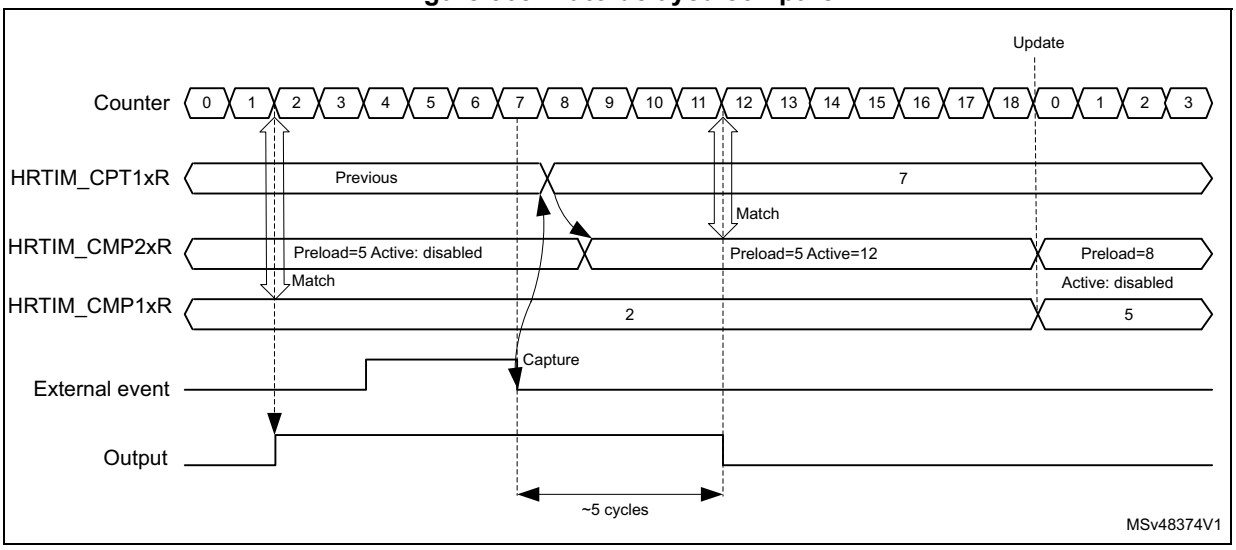
- DELCMPx[1:0] = 10
- DELCMPx[1:0] = 00
- DELCMPx[1:0] = 11

As an example, *Figure 365* shows how the following signal is generated:

- Output set when the counter is equal to compare 1 value
- Output reset 5 cycles after a falling edge on a given external event

Note: To simplify the figure, the high-resolution is not used in this example (CKPSC[2:0] = 101), thus the counter is incremented at the f_{HRTIM} rate. Similarly, the external event signal is shown without any resynchronization delay: practically, there is a delay of one to two f_{HRTIM} clock periods between the falling edge and the capture event due to an internal resynchronization stage which is necessary to process external input signals.

Figure 365. Auto-delayed compare



A regular compare channel (e.g. compare 1) is used for the output set: as soon as the counter matches the content of the compare register, the output goes to its active state.

A delayed compare is used for the output reset: the compare event can be generated only if a capture event has occurred. No event is generated when the counter matches the delayed compare value (counter = 4). Once the capture event has been triggered by the external event, the content of the capture register is summed to the delayed compare value to have the new compare value. In the example, the autodelayed value 4 is summed to the capture equal to 7 to give a value of 12 in the autodelayed compare register. From this time on, the compare event can be generated and happens when the counter is equal to 12, causing the output to be reset.

Overcapture management in autodelayed mode

Overcapture is prevented when the autodelayed mode is enabled (DELCMPx[1:0] = 01, 10, 11).

When multiple capture requests occur within the same counting period, only the first capture is taken into account to compute the autodelayed compare value. A new capture is possible only:

- Once the autodelayed compare has matched the counter value (compare event)
- Once the counter has rolled over (period)
- Once the timer has been reset

Changing autodelayed compare values

When the autodelayed compare value is preloaded (PREEN bit set), the new compare value is taken into account on the next coming update event (for instance on the period event), regardless of when the compare register was written and if the capture occurred (see *Figure 365*, where the delay is changed when the counter rolls over).

When the preload is disabled (PREEN bit reset), the new compare value is taken into account immediately, even if it is modified after the capture event has occurred, as per the example below:

1. At t1, DELCMP2 = 1.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, a capture event occurs capturing the value CPTR1 = 0x20. => comparison enabled, compare value = 0x60
4. At t4, CMP2_act = 0x100 (before the counter reached value CPTR1 + 0x40) => comparison still enabled, new compare value = 0x120
5. At t5, the counter reaches the period value => comparison disabled, cmp2_act = 0x100

Similarly, if the CMP1(CMP3) value changes while DELCMPx = 10 or 11, and preload is disabled:

1. At t1, DELCMP2 = 2.
2. At t2, CMP2_act = 0x40 => comparison disabled
3. At t3, CMP3 event occurs - CMP3_act = 0x50 before capture 1 event occurs => comparison enabled, compare value = 0x90
4. At t4, CMP3_act = 0x100 (before the counter reached value 0x90) => comparison still enabled, compare 2 event occurs at = 0x140

Triggered-half mode

The purpose of this mode is to allow the synchronization of two interleaved converters that have variable frequency operation, and require a 180° phase-shift. The basic principle is to

have a master-slave system. The slave converter synchronization is continuously adjusted based on the previous switching period of the master converter.

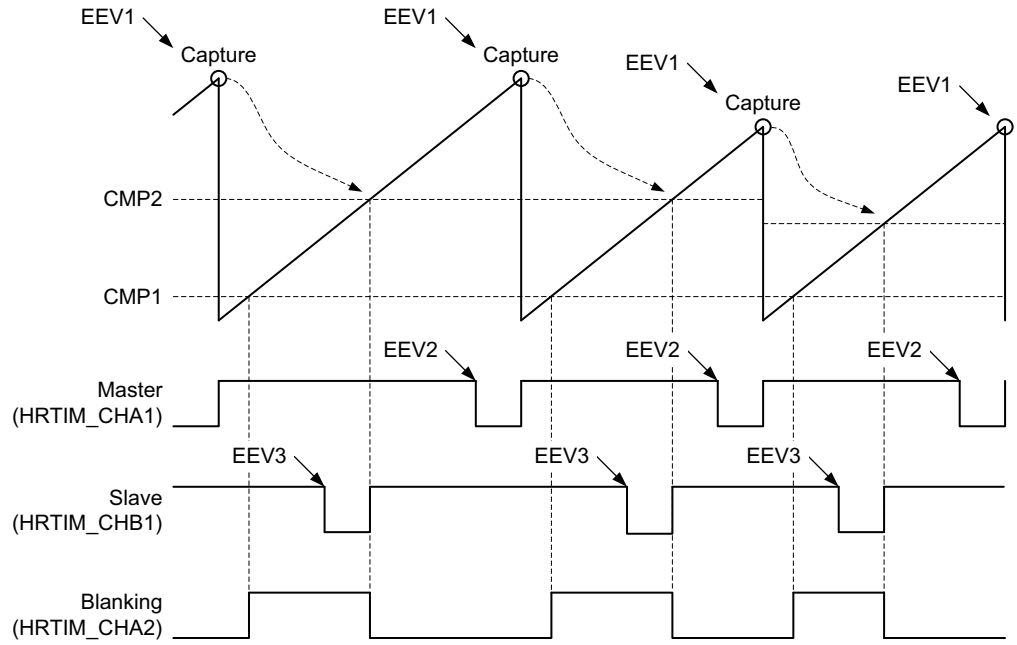
This is done using the capture unit. The switching period of the master converter is captured, divided by two and then stored in the compare 2 register by hardware. The compare 2 register contains a value equal to half of the captured period, which is the master converter switching period. The compare 2 event can then be used to trigger a second timing unit that manages the slave converter.

This mode is enabled by setting the TRGHLF bit in the HRTIM_TIMxCR2 register. This bit cannot be changed once the timer is operating (TxEN bit set).

The triggered-half mode must not be used simultaneously with other modes using CMP2 (dual channel DAC trigger, interleaved and balanced idle modes).

The initial value CMP2 can be written by the user, but is ignored once the first capture is triggered. The preload mechanism is disabled for CMP2 when the TRGHLF bit is reset.

Figure 366. Triggered-half mode example



HRTIM_CHA1
Set on EEV1 (EEV1 triggers Capture1)
Reset on EEV2
EEV1 Blanking on TIMA_CMP1 (avoid frequency run-away)
HRTIM_CHB1
Set on CMP2 (triggered-half) or EEV1 blanking by TIMA_TA2
(TB1 is set if EEV1 occurs between CMP1 and CMP2, else on CMP2)
Reset on EEV3
HRTIM_CHA2
Set on TA_CMP1
Reset on TA_CMP2

MSv45796V3

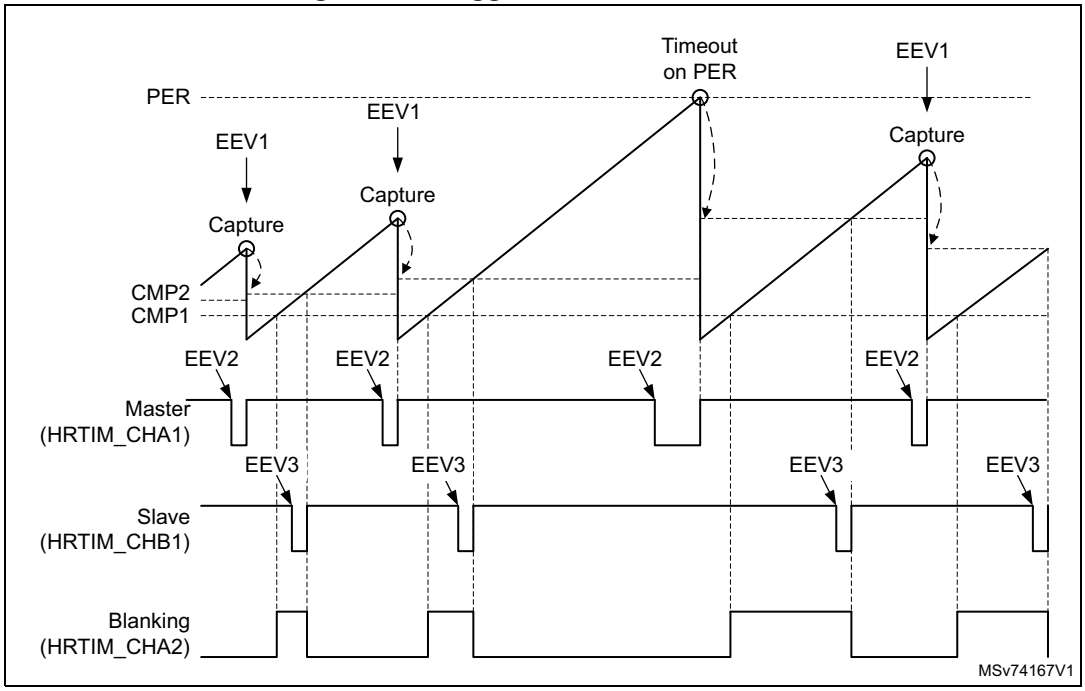
Note: In triggered half mode, the compare2 register is controlled by hardware and writing it has no effect. However the written value is stored in the preload register and becomes active on the update event following the exit of this mode.

It is possible to generate a timeout event when the external event is missing, as shown on *Figure 367* below. This mode is enabled with the TRGHLFT in HRTIM_CR2 register.

No capture event is generated when a timeout occurs. A timeout situation can be detected using the PER interrupt.

The timeout occurs regardless of the repetition value.

Figure 367. Triggered-half mode timeout



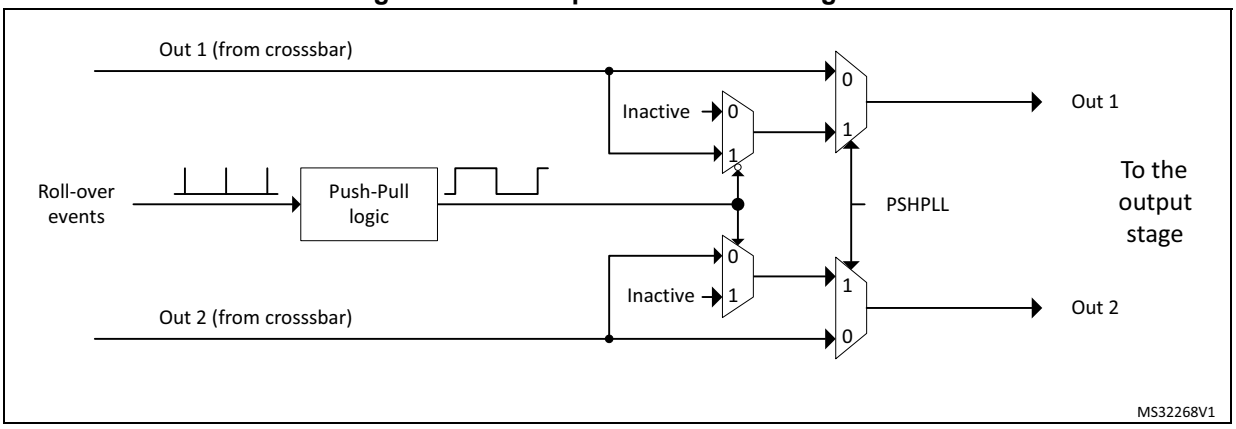
Push-pull mode

This mode primarily aims at driving converters using push-pull topologies. It also needs to be enabled when the delayed idle protection is required, typically for resonant converters (refer to *Section 40.3.10: Delayed protection*).

The push-pull mode is enabled by setting the PSHPLL bit in the HRTIM_TIMxCR register.

It applies the signals generated by the crossbar to output 1 and output 2 alternatively, on the period basis, maintaining the other output to its inactive state. The redirection rate (push-pull frequency) is defined by the timer's period event, as shown on *Figure 368*. The push-pull period is twice the timer counting period.

Figure 368. Push-pull mode block diagram



The push-pull mode is available when the timer operates in continuous mode and in single-shot mode. It is necessary to disable the timer to stop a push-pull operation and to reset the counter before reenabling it.

To get a correct behavior, the event selected as source of the counter reset must be also selected for setting (or resetting) the output. It must set the output if the output is set on period, else it must reset it. If it is not done, the output switching from its inactive period to its active period may be incorrect (may unexpectedly rise or may unexpectedly stay low).

The signal shape is defined using HRTIM_SETxyR and HRTIM_RSTxyR for both outputs. It is necessary to have HRTIM_SETx1R = HRTIM_SETx2R and HRTIM_RSTx1R = HRTIM_RSTx2R to have both outputs with identical waveforms and to achieve a balanced operation. Still, it is possible to have different programming on both outputs for other uses.

The CPPSAT status bit in HRTIM_TIMxISR indicates on which output the signal is currently active. CPPSTAT is reset when the push-pull mode is disabled.

In the example given on *Figure 369*, the timer internal waveform is defined as follows:

- Output set on period event
- Output reset on compare 1 match event

Figure 369. Push-pull mode example

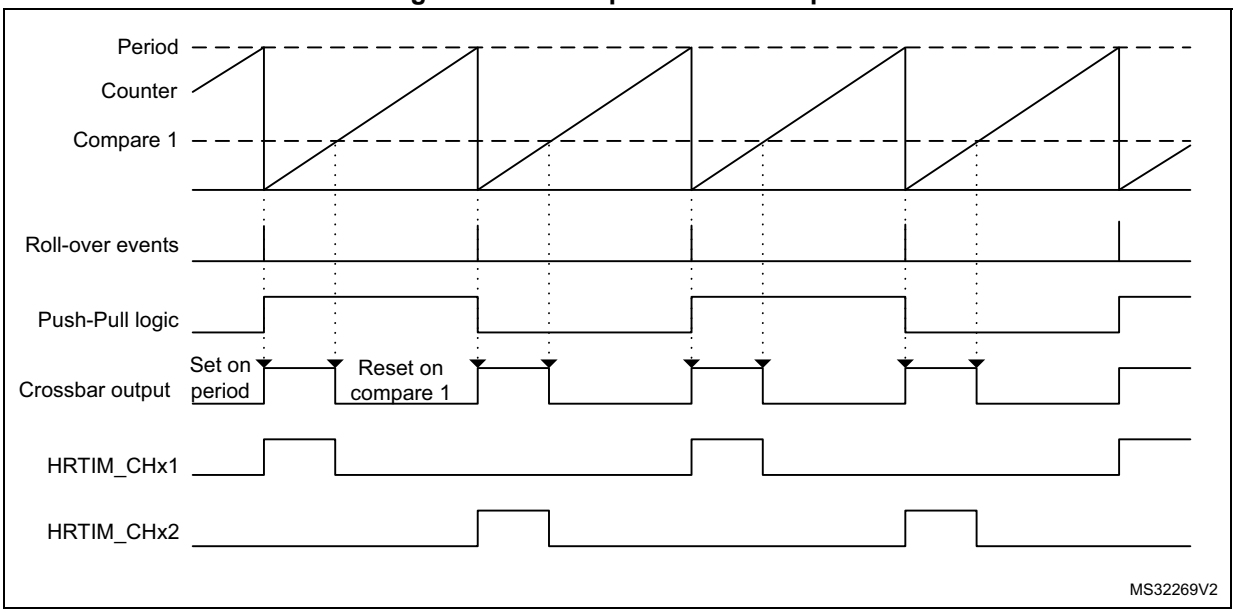
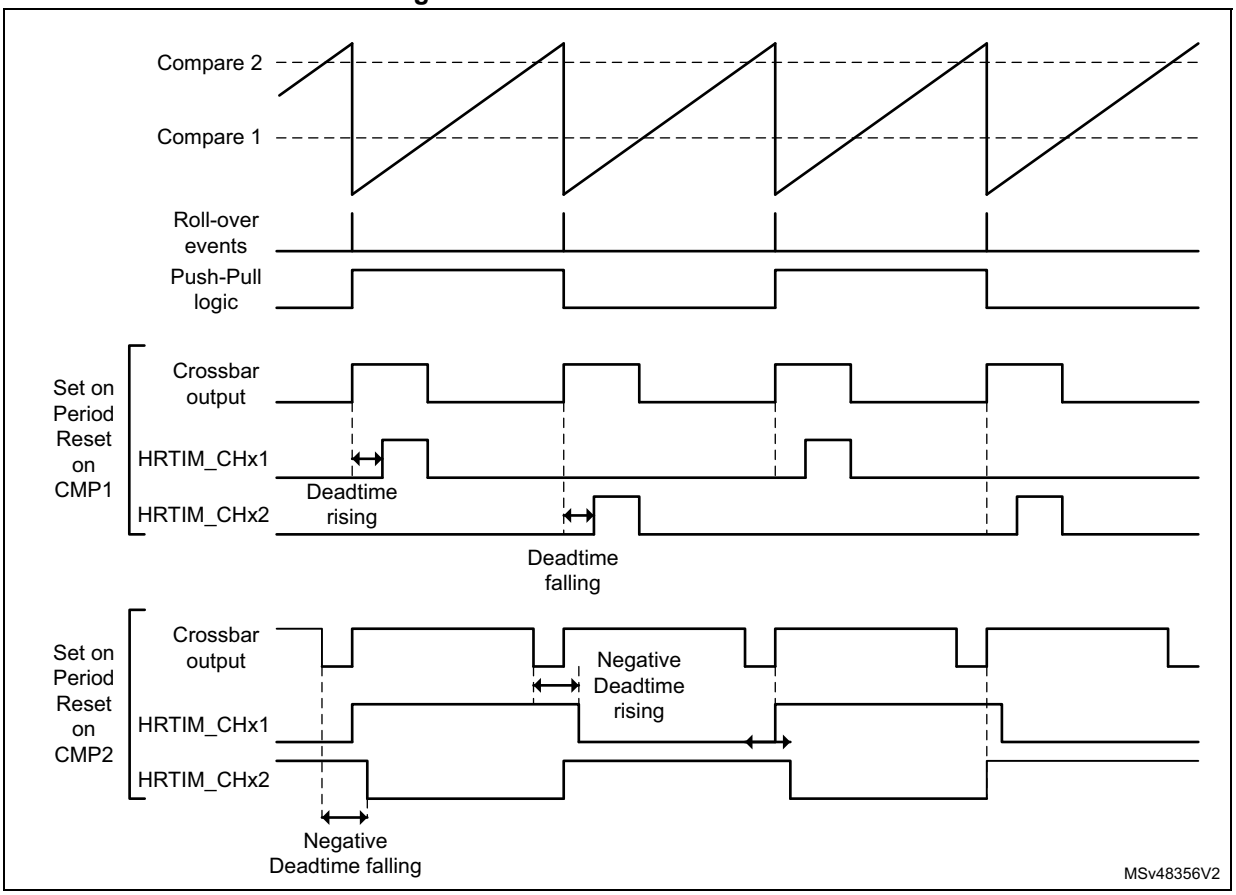


Figure 370 shows how the deadtime is inserted in push-pull mode, for both positive and negative deadtimes. In this case, the outputs are not complemented anymore, and the deadtime is applied on each output individually (both ouput 1 and output 2 of the crossbar are used).

Figure 370. Push-Pull with deadtime

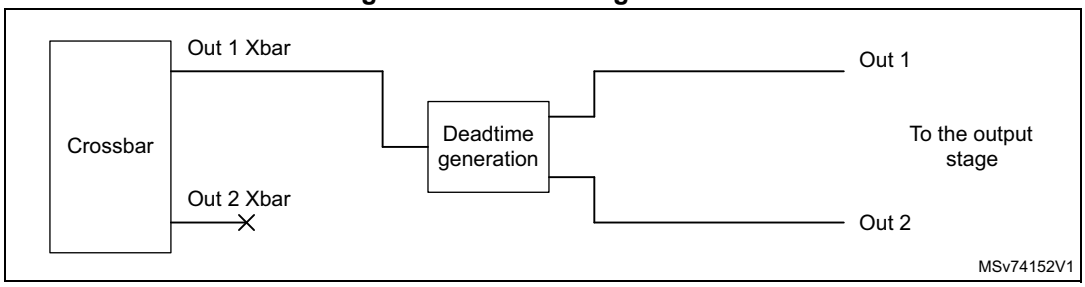


Deadtime

A deadtime unit is used to generate a couple of complementary signals from a single reference waveform, with programmable delays between active state transitions. This is commonly used for topologies using half-bridges or full bridges. It simplifies the software: only one waveform is programmed and controlled to drive two outputs.

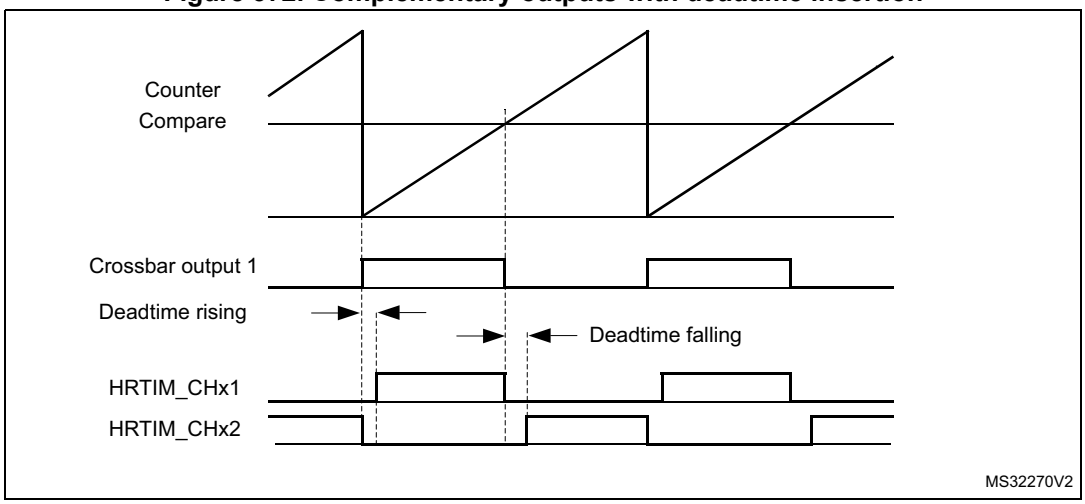
The deadtime is enabled by setting DTEN bit in HRTIM_OUTxR register. The complementary signals are built based on the reference waveform defined for output 1, using HRTIM_SETx1R and HRTIM_RSTx1R registers: HRTIM_SETx2R and HRTIM_RSTx2R registers are not significant when DTEN bit is set, as shown on *Figure 371*.

Figure 371. Deadtime generation



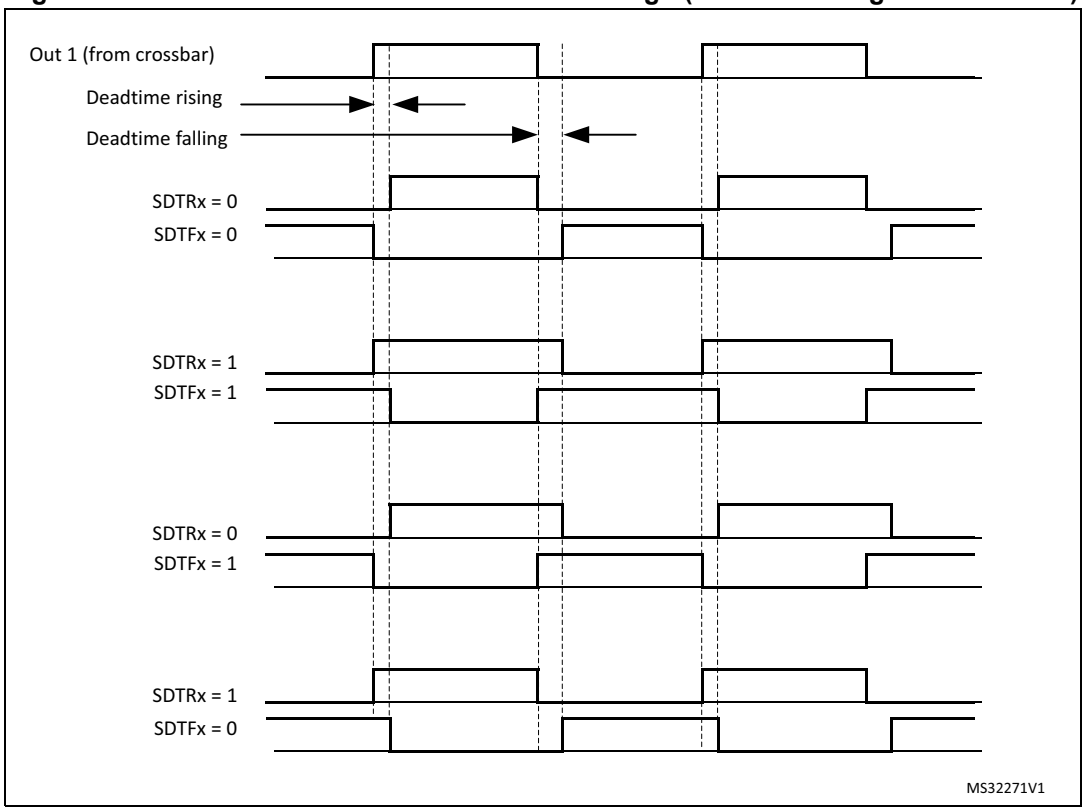
Two deadtimes can be defined in relationship with the rising edge and the falling edge of the reference waveform, as in *Figure 372*.

Figure 372. Complementary outputs with deadtime insertion



Negative deadtime values can be defined when some control overlap is required. This is done using the deadtime sign bits (SDTFx and SDTRx bits in HRTIM_DTxR register). *Figure 373* shows complementary signal waveforms depending on respective signs.

Figure 373. Deadtime insertion versus deadtime sign (1 indicates negative deadtime)



The deadtime values are defined with DTFx[8:0] and DTRx[8:0] bitfields and based on a specific clock prescaled according to DTPRSC[2:0] bits, as follows:

$t_{DTx} = \pm DTx[8:0] \times t_{DTG}$

where x is either R or F and $t_{DTG} = (2^{(DTPRSC[2:0])}) \times (t_{HRTIM} / 8)$.

Table 425 gives the resolution and maximum absolute values depending on the prescaler value.

Table 425. Deadtime resolution and max absolute values

DTPRSC[2:0]	t_{DTG}	t_{DTx} max	f_{HRTIM}= 280 MHz	
			t_{DTG} (ns)	$\|t_{DTx}\|$ max (µs)
000	$t_{HRTIM} / 8$	511 * t_{DTG}	0.45	0.23
001	$t_{HRTIM} / 4$		0.9	0.46
010	$t_{HRTIM} / 2$		1.8	0.93
011	t_{HRTIM}		3.6	1.86
100	$2 \times t_{HRTIM}$		7.1	3.71
101	$4 \times t_{HRTIM}$		14.3	7.43
110	$8 \times t_{HRTIM}$		28.6	14.86
111	$16 \times t_{HRTIM}$		57.1	29.71

Figure 374 to *Figure 377* present how the deadtime generator behaves for reference waveforms with pulsewidth below the deadtime values, for all deadtime configurations.

Figure 374. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 0)

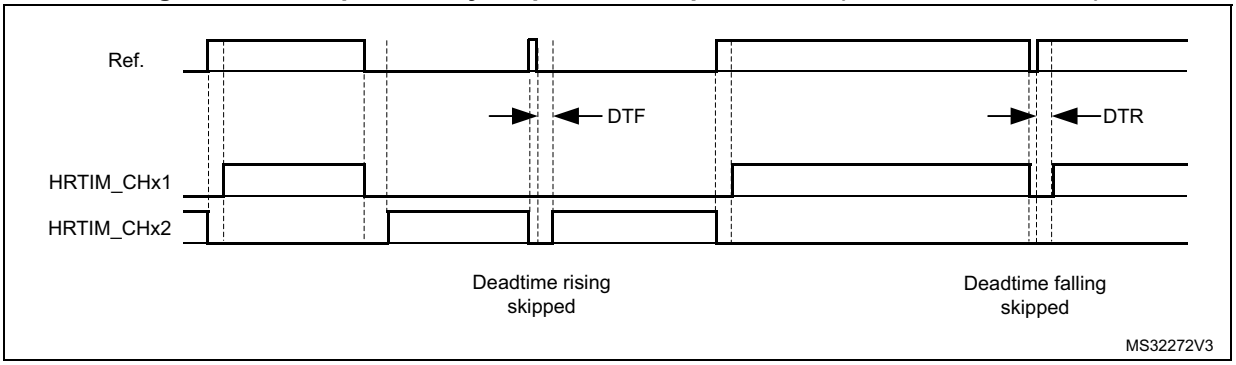


Figure 375. Complementary outputs for low pulsewidth (SDTRx = SDTFx = 1)

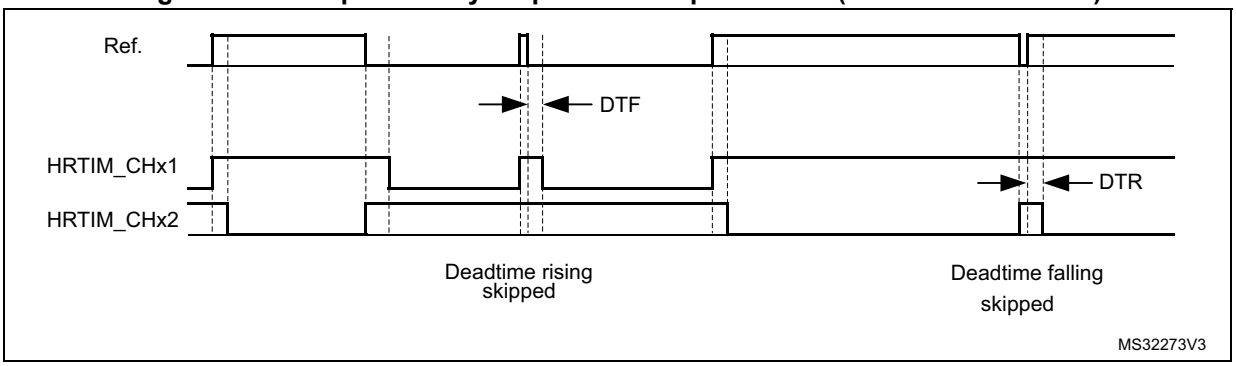


Figure 376. Complementary outputs for low pulsewidth (SDTRx = 0, SDTFx = 1)

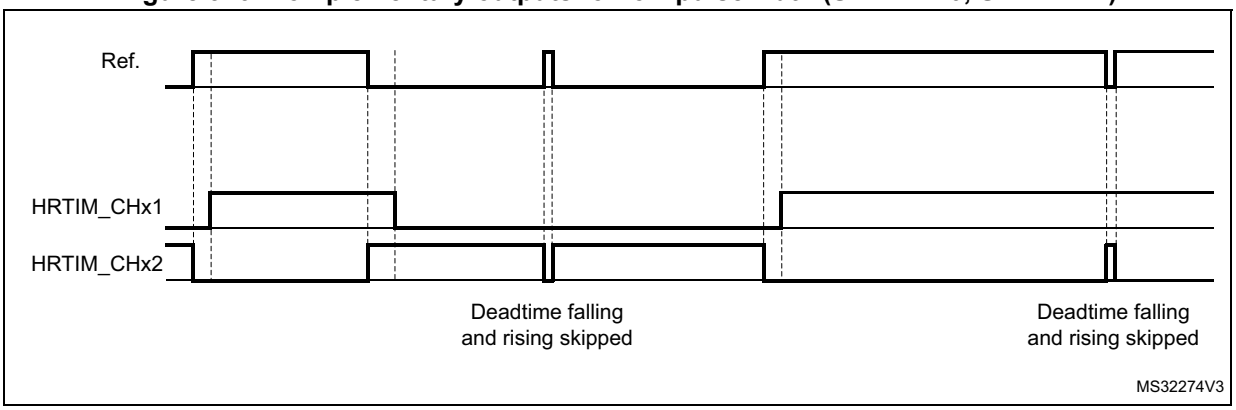
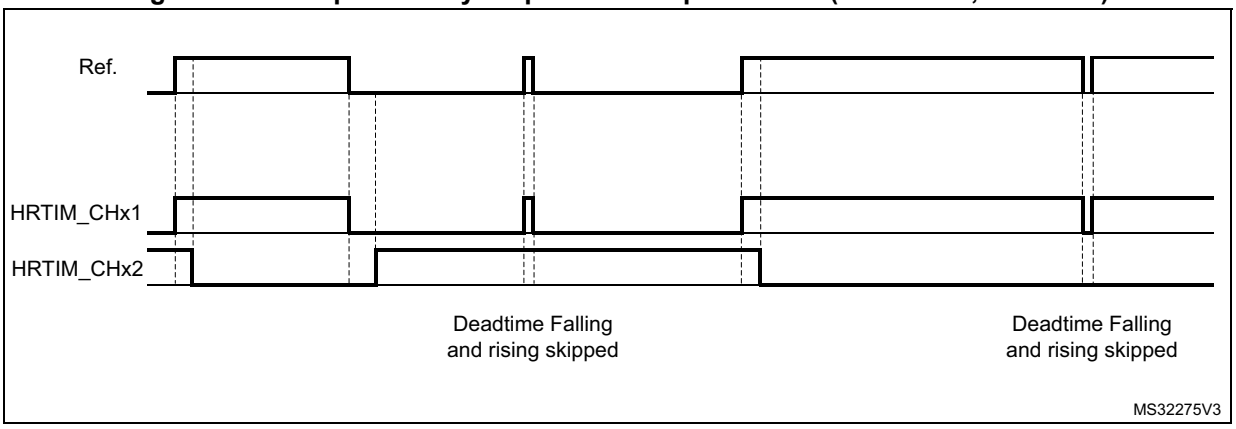


Figure 377. Complementary outputs for low pulsewidth (SDTRx = 1, SDTFx=0)



For safety purposes, it is possible to prevent any spurious write into the deadtime registers by locking the sign and/or the value of the deadtime using DTFLKx, DTRLKx, DTFSLKx, and DTRSLKx. Once these bits are set, the related bits and bitfields are becoming read only until the next system reset.

The DTEN bit is preloaded, and can be written while the timer is enabled.

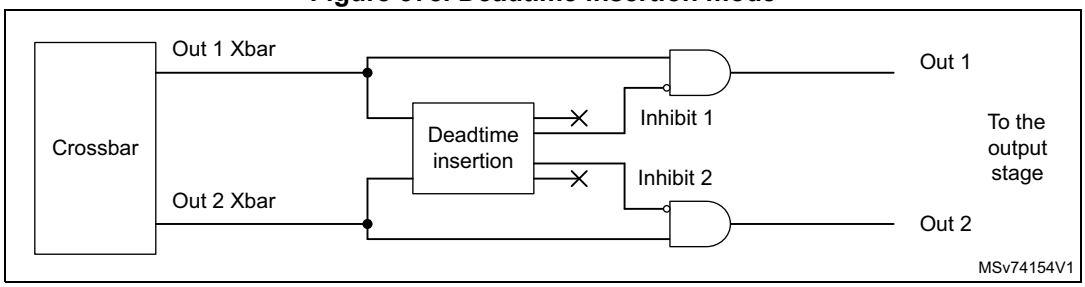
Caution: DTEN bit must not be changed when the timer outputs are set/reset by another timer (while TxEN is reset)

Note: For the particular case where DTEN must be set while the burst mode is enabled with a deadtime upon entry (BME = 1, DIDL = 1, IDLEM = 1), it is necessary to force the two outputs in their IDLES state by software commands (SST, RST bits) before setting DTEN bit. This is to avoid any side effect resulting from a burst mode entry that would happen immediately before a deadtime enable.

Deadtime insertion modes

Two alternative deadtime options are available to handle situations where the signals are not complementary. In this case, the output 1 and output 2 signals are independently defined, and a deadtime insertion will only prevent any deadtime violation, as show on *Figure 378* below.

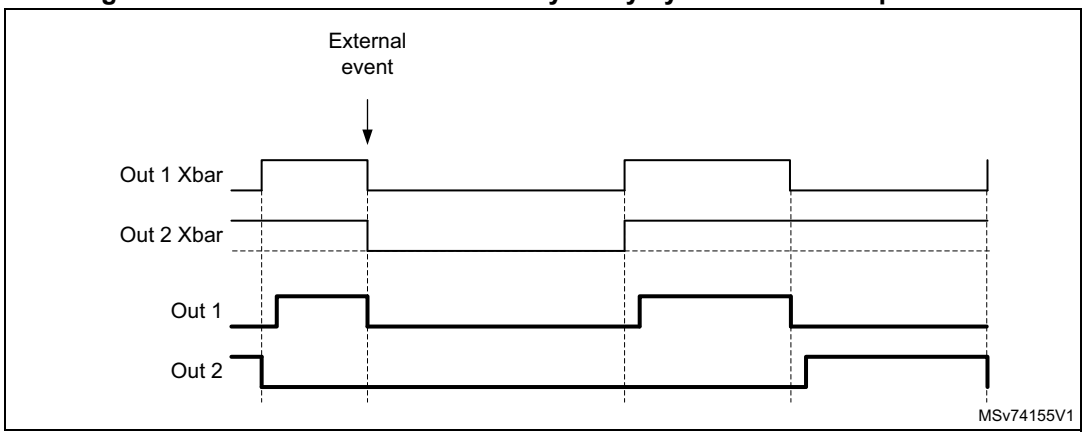
Figure 378. Deadtime insertion mode



The deadtime insertion mode is enabled by setting DTM[1:0] = 01 in the HRTIM_DTxR register.

The example on *Figure 379* below shows an example of waveform in deadtime insertion mode, with a cycle-by-cycle over-current protection. In this case, the output 2 is set on period and reset on external event.

Figure 379. Deadtime insertion with cycle-by-cycle overcurrent protection

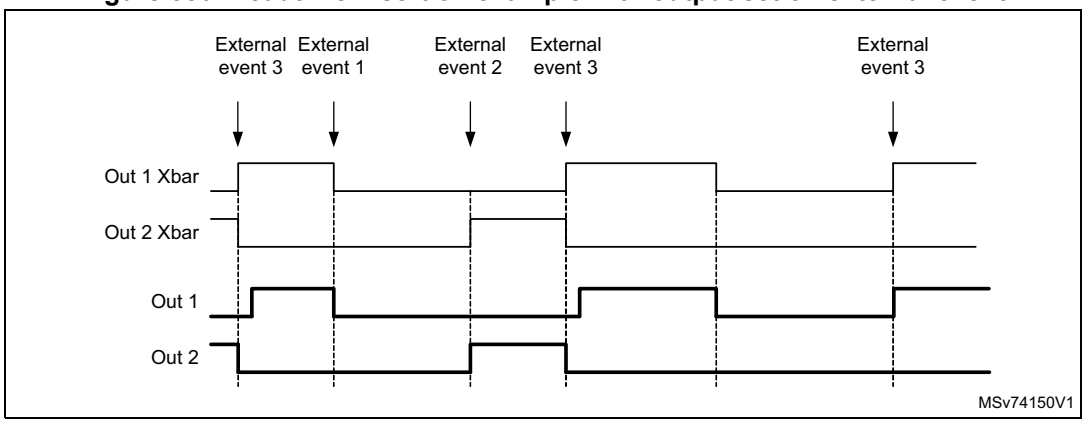


The example on *Figure 380* below shows an example of waveform in deadtime insertion mode, with the following conditions:

- output 1 set on external event 3, reset on external event 1
- output 2 set on external event 2, reset on external event 3
- counter reset on external event 3

It must be noticed that output 2 is set immediately after external event 2, without any deadtime delay, if the output 1 is inactive.

Figure 380. Deadtime insertion example with output set on external event

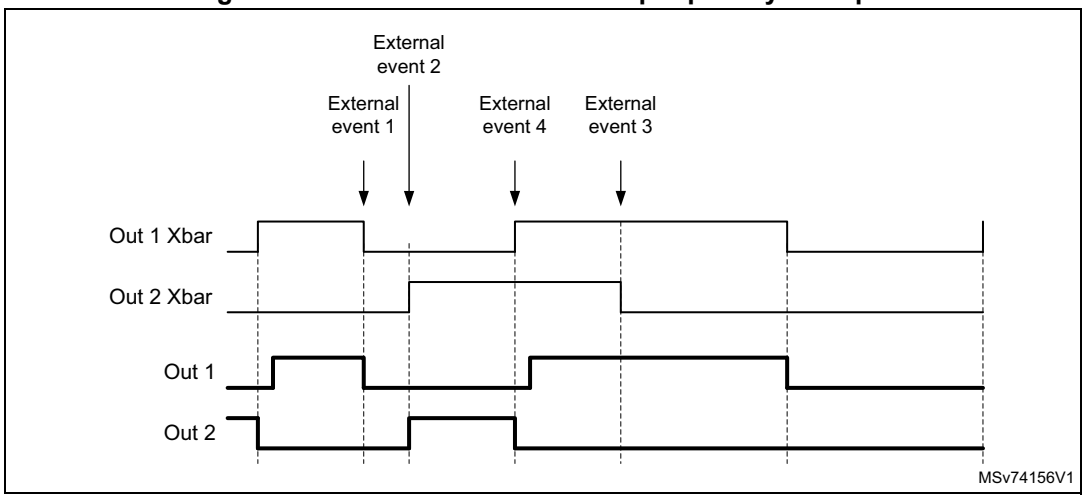


The example on *Figure 380* below shows an example of waveform in deadtime insertion mode, with the following conditions:

- output 1 set on external event 4, reset on external event 1
- output 2 set on external event 2, reset on external event 3

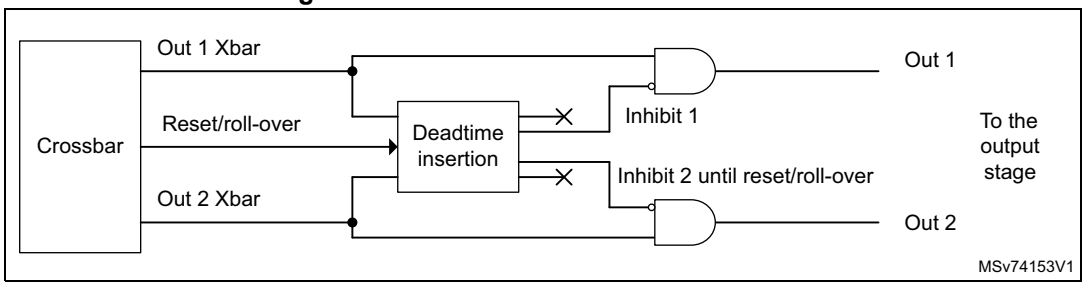
It must be noticed that output 1 set event has priority over output 2 active state.

Figure 381. Deadtime insertion: output priority example



The deadtime insertion and disarm mode is enabled by setting DTM[1:0] = 10 in the HRTIM_DTxR register, and prevent the output 2 to be set again until the next reset/roll-over event, as shown on *Figure 382* and *Figure 383* below.

Figure 382. Deadtime insertion and disarm



The conditions to disarm the output 2 (i.e. lock output 2 and ignore any other set events) are the following:

- One (or multiple) external event(s) selected to reset the output 1
- At least one external event enabled

Note: If the output 2 crossbar is active before output 1 is disabled (output 1 and output 2 overlapping), the disarming logic will not reset output 2. If output 2 must be disabled in these conditions (i.e. external event 2 occurring before External event 1, it is necessary to program the same external event for resetting both Out1 and Out2.

Figure 383. Deadtime insertion and disarm example 1

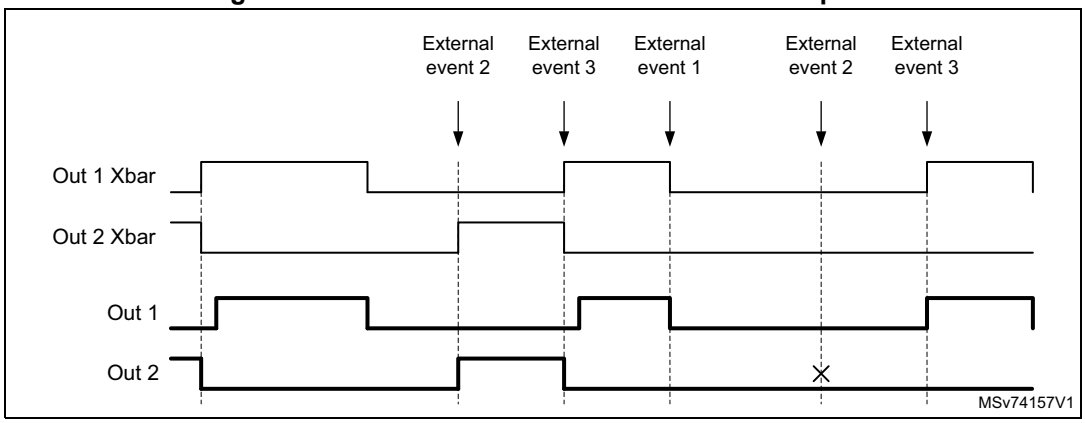
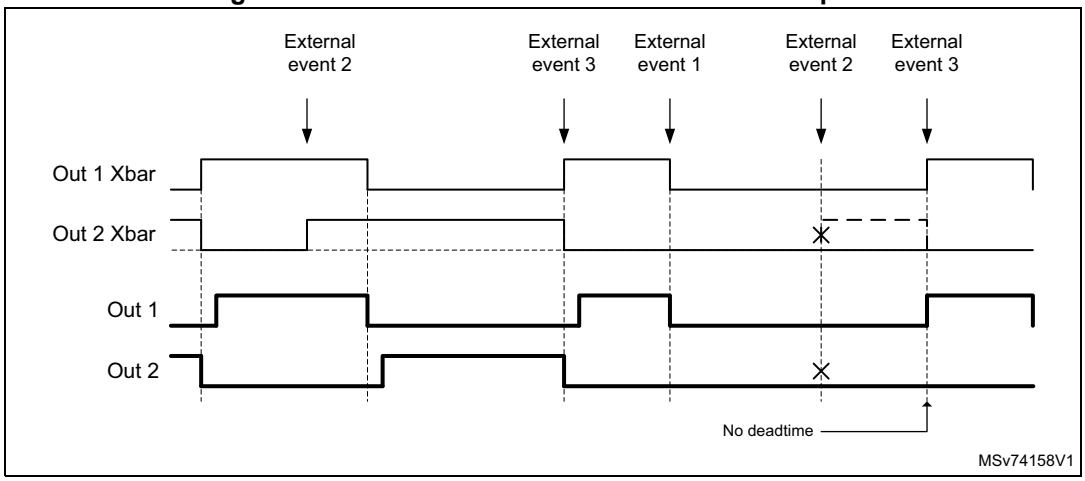


Figure 384 shows that there is no deadtime insertion when an output is set while the deadtime insertion period is over (i.e. output 2 set immediately on External Event 3).

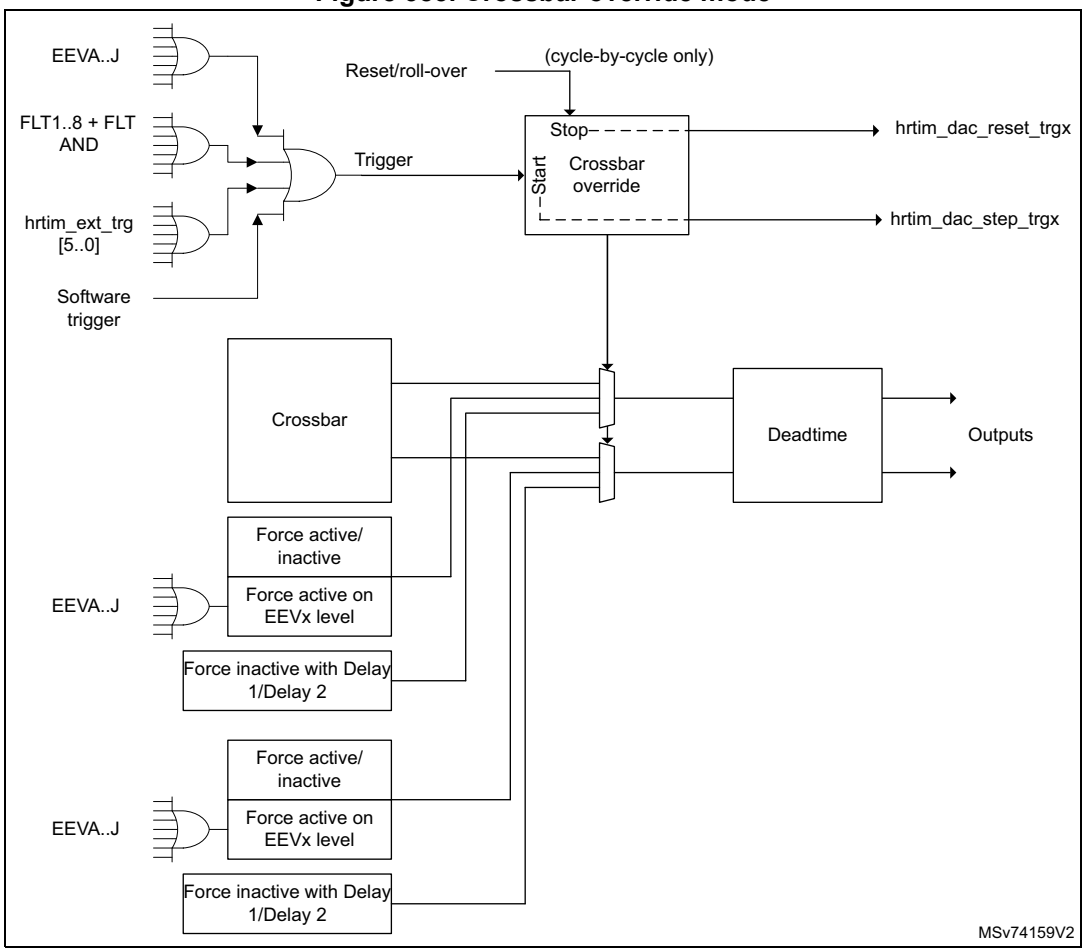
Figure 384. Deadtime insertion and disarm example 2



Crossbar override

This mode gives the possibility to override the output crossbar programming and take over the output control to force a level based on a fixed setting or based on a given external event or fault, as shown on *Figure 385* below.

Figure 385. Crossbar override mode



This mode is triggered on one or multiple sources selected in HRTIM_COTRGxR register:

- Software trigger
- Fault 1..8 and AND combined faults inputs
- External events A..J
- Six external crossbar override triggers hrtim_ext_trg[5:0] (hrtim_ext_trg[7:6] are not available as triggers)

Once the crossbar override mode is entered, specific external events will force high or low the outputs when active.

The crossbar override logic provides 2 options:

- Latched: once triggered, the crossbar override mode is maintained until a software intervention to disengage it.
- Cycle-by-cycle mode: the crossbar override mode exits automatically and normal operation is resumed by hardware. Typically, the system will re-arm once the over-

current condition has disappeared.
This mode is only applicable in force on event mode (COFMx[2:0] = 100).

In cycle-by-cycle mode (COM bit reset in HRTIM_TIMxCR2), the Crossbar override function is disengaged when both:

- The trigger event is inactive
- The next reset/roll-over happens (conditioned by ROM[1:0] in center-aligned mode)

In latched mode (COM bit set in HRTIM_TIMxCR2), the crossbar override mode can be disengaged in two ways:

- With the rearm bit COR in HRTIM_TIMxCR2. The rearming will be effective only of the trigerring condition has disappeared (status can be polled using the COTS bit in the HRTIM_TIMxISR register).
- With the COE bit reset, to disable it permanently.

The DAC values can be updated synchronously with the crossbar override mode triggering. The triggers are generated when the DCDCOE (Dual Channel DAC Crossbar Override Enable) bit is set in HRTIM_TIMxCR2 (the DCDE value is overridden when DCDCOE=1).

- hrtim_dac_reset_trgx triggers are generated only in cycle-by-cycle mode when the crossbar override stops, and DCDR setting applies.
- hrtim_dac_step_trgx triggers are generated when the crossbar override starts.

The crossbar override mode provides several options, defined independently for each output with the COFMA[2:0] and COFMB[2:0] bitfields in the HRTIM_COFExR registers.

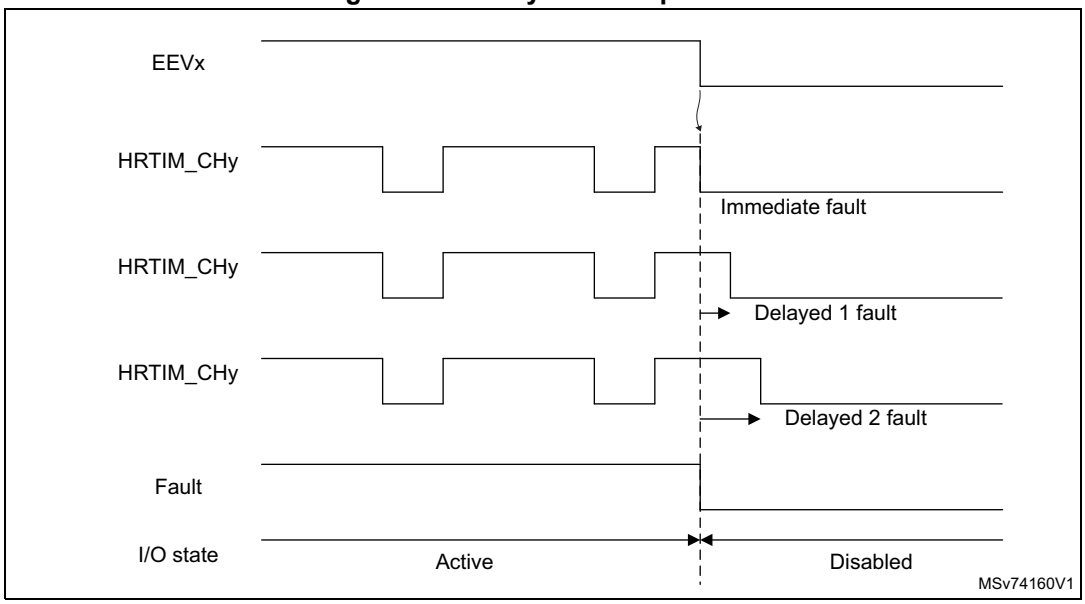
Delayed fault operation

This mode is enabled with COFMx[2:0] = 000 and 001. When the crossbar override mode is triggered, the output level is maintained as it is and forced to the inactive state after a programmable duration. The purpose of this mode is have a delayed shutdown sequence for multi-level inverters. 2 delays can be selected for a given output: Delay 1 with COFMx[2:0] = 000 or Delay 2 with COFMx[2:0] = 001, as shown on *Figure 386*.The delay 1/2 durations are defined with the DLYxFLTD[8:0] bitfields in the HRTIM_CODExR register.

This mode can also be used with deadtime disabled (DTEN=0).

Warning: It is mandatory to program a Delay 2 duration above the Delay 1 duration: DLY2FLTD[8:0] > DLY1FLTD[8:0].

Figure 386. Delayed fault operation

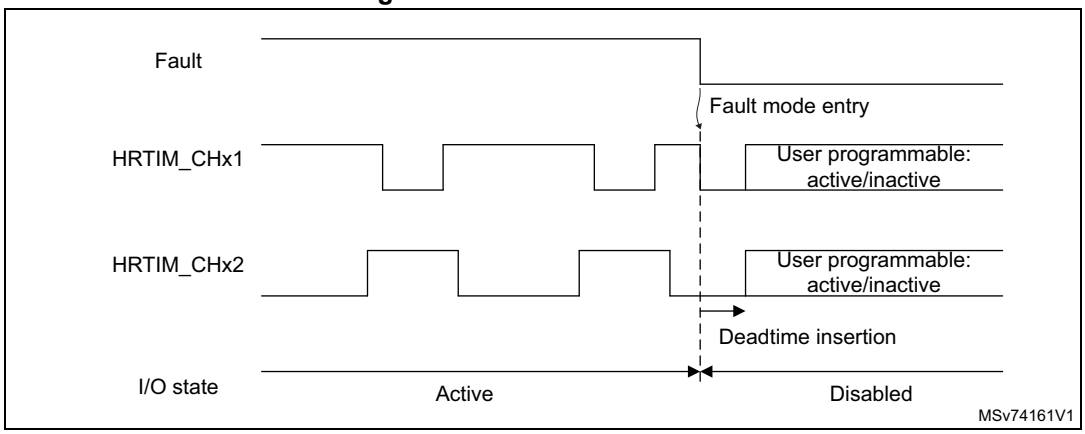


Deadtime on Fault

This mode is enabled with COFMx[2:0] = 010 and 011. When the crossbar override mode is triggered, the output level is forced to an active or inactive state, as shown on *Figure 387* below. As the crossbar override logic is placed prior to the deadtime insertion logic, this mechanism guarantees that an output can be force active without any risk of deadtime violation. If both outputs are forced to the active state, both output 1 and output 2 are forced inactive.

This mode cannot be used with deadtime disabled (DTEN = 0).

Figure 387. Deadtime on Fault

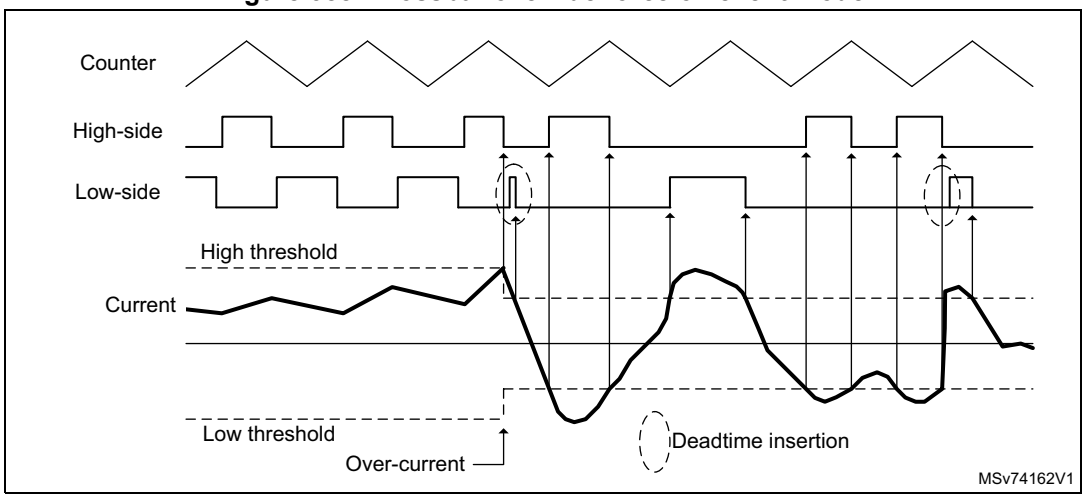


Force on event

This mode is enabled with COFMx[2:0] = 100. When the crossbar override mode is triggered, the output level is forced active on the event selected in the EEVxFRCA and EEVxFRCB bits, in the HRTIM_COFExR register, as shown on *Figure 388* below.

This mode cannot be used with deadtime disabled (DTEN = 0).

Figure 388. Crossbar override force on event mode



Status bits and interrupts

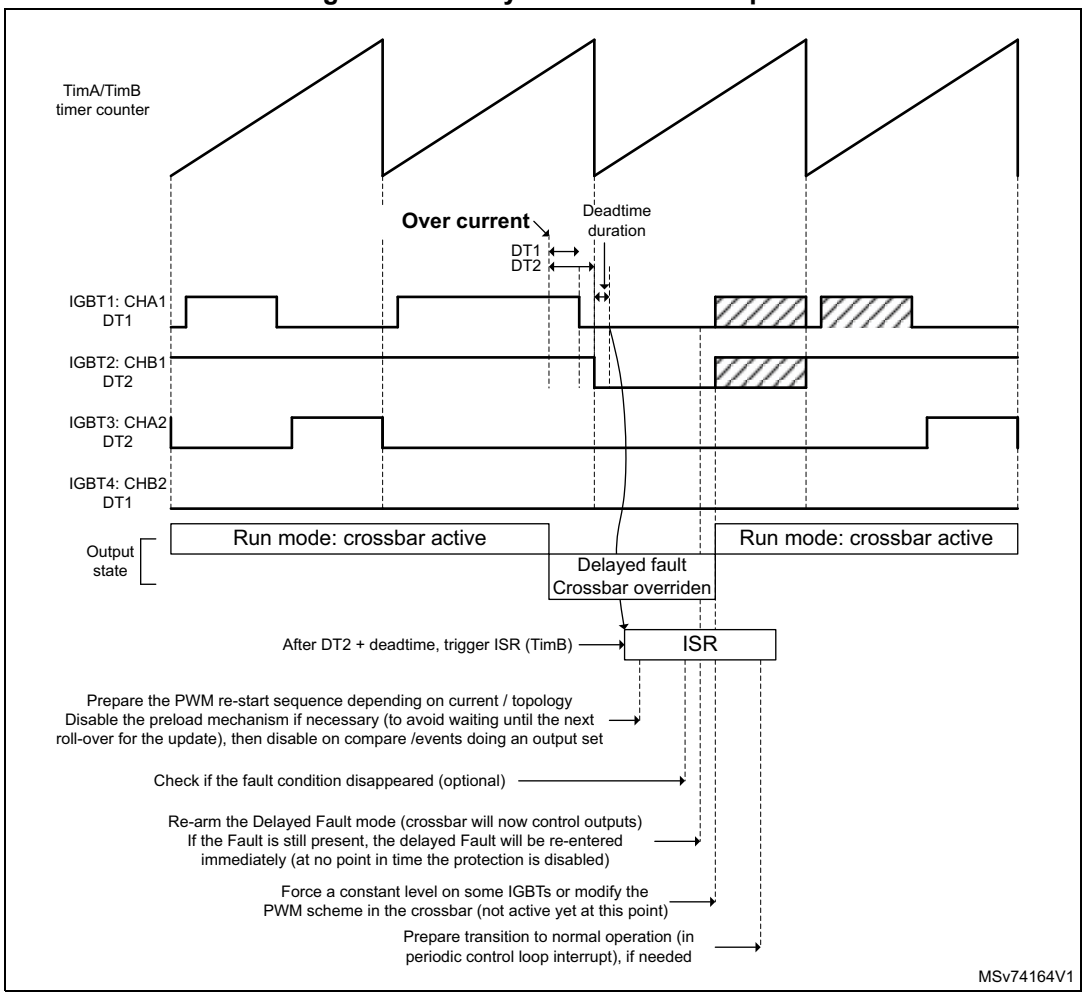
Two status bits are available for software polling, in the HRTIM_TIMxISR register. The COS bit indicates if the crossbar override mode is active, or if the regular crossbar operation is active or was resumed.The COTS bit indicates if the crossbar override trigger is active or not, and can be polled before rearming the crossbar override mode. Both bits are only status bits.

Two interrupts are available. The crossbar override trigger interrupt indicates that regular crossbar operation was interrupted and crossbar override mode was started (COT interrupt flag in HRTIM_TIMxISR). The crossbar override ream delay elapsed interrupt (CORDE interrupt flag in HRTIM_TIMxISR) indicates that the regular crossbar mode can be rearmed. This flag is raised once the Delay 2 period plus the deadtime duration is elapsed, to indicate that the crossbar override delayed fault entry sequence is completed before rearming.

Any rearming attempt before CORDE interrupt is ignored. The *Figure 389* below gives an example of a delayed fault exit sequence [To be completed and reworked].

This CORDE interrupt is generated only when at least on of the one of the 2 channels is configured to Delayed fault force modes (COFMx[2:0] = 000 and 001).

Figure 389. Delayed Fault exit example

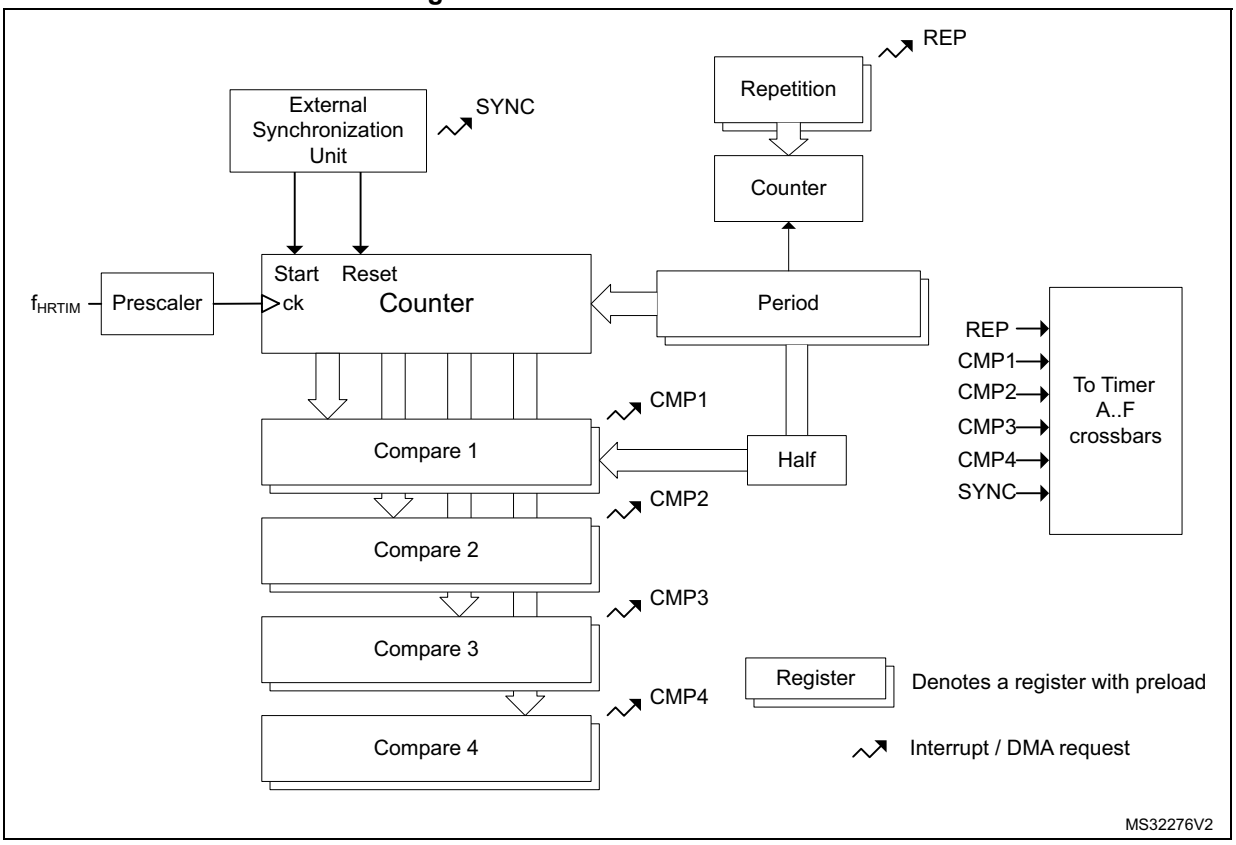


40.3.5 Master timer

The main purpose of the master timer is to provide common signals to the 6 timing units, either for synchronization purpose or to set/reset outputs. It does not have direct control over any outputs, but still can be used indirectly by the set/reset crossbars.

Figure 390 provides an overview of the master timer.

Figure 390. Master timer overview



The master timer is based on the very same architecture as the timing units, with the following differences:

- It does not have outputs associated with, nor output related control
- It does not have its own crossbar unit, nor push-pull or deadtime mode
- It can only be reset by the external synchronization circuitry
- It does not have a capture unit, nor the autodelayed mode
- It does not include external event blanking and windowing circuitry
- It has a limited set of interrupt/DMA requests: compare 1..4, repetition, register update and external synchronization event.

The master timer control register includes all the timer enable bits, for the master and timer A..F timing units. This allows to have all timer synchronously started with a single write access.

It also handles the external synchronization for the whole HRTIM timer (see *Section 40.3.19: Synchronizing the HRTIM with other timers or HRTIM instances*), with both MCU internal and external (inputs/outputs) resources.

Master timer control registers are mapped with the same offset as the timing units' registers.

40.3.6 Up-down counting mode

The HRTIM is natively designed with up-counters. It offers however an operating mode with up-down counters, also called center-aligned mode.

This mode is enabled using the UDM bit in the HRTIM_TIMxCR2 register. This bit must not be changed once the timer is operating (TxEN bit set). It is only available for the TIMA..F. The master timer only works in up-counting mode.

Not all HRTIM features are supported in up-down counting. This section details the functional differences versus up-counting mode.

The period in HRTIM_PERxR must be preloaded (or static) in up-down mode. It can be updated only on period event or on counter reset.

The set/reset crossbar programming differs as follows:

The events coming from the timing units are setting/resetting the outputs depending on the counter up/down direction:

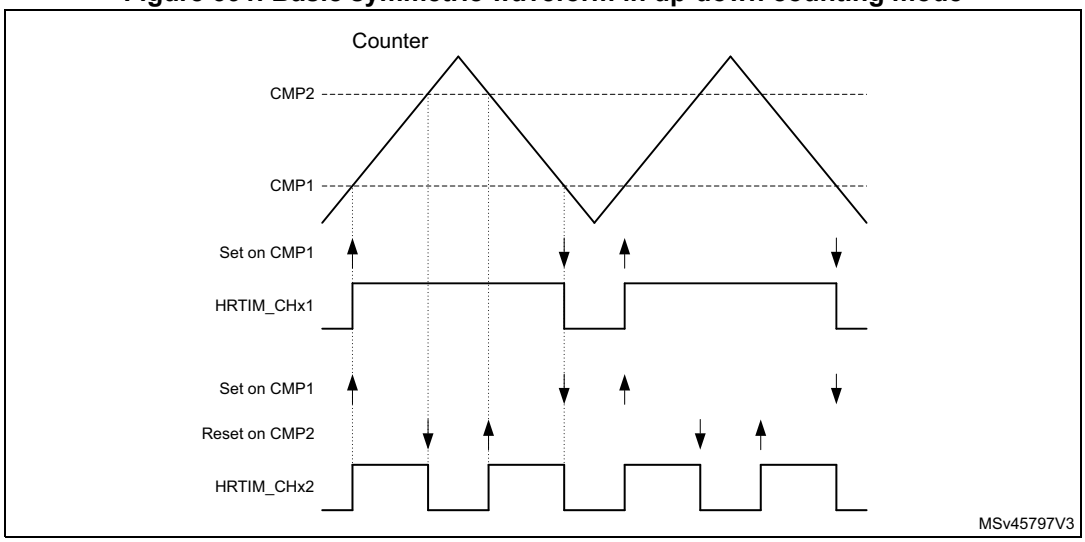
- If the event is enabled in the HRTIM_SETxyR register, it sets the output during up-counting and reset it during down-counting.
- If the event is enabled in the HRTIM_RSTxyR register, it resets the output during up-counting and set it during down-counting.
- If the events are enabled both in HRTIM_SETxyR and HRTIM_RSTxyR registers, the output toggles.

This applies to:

- The timing unit: period, compare 1..4, register update (6 events)
- The master timer: period, compare 1..4, HRTIM synchronization (6 events)
- All other timing units (e.g. timer B..F for timer A): TIMEVNT1..9 (9 events described in *Table 422*).

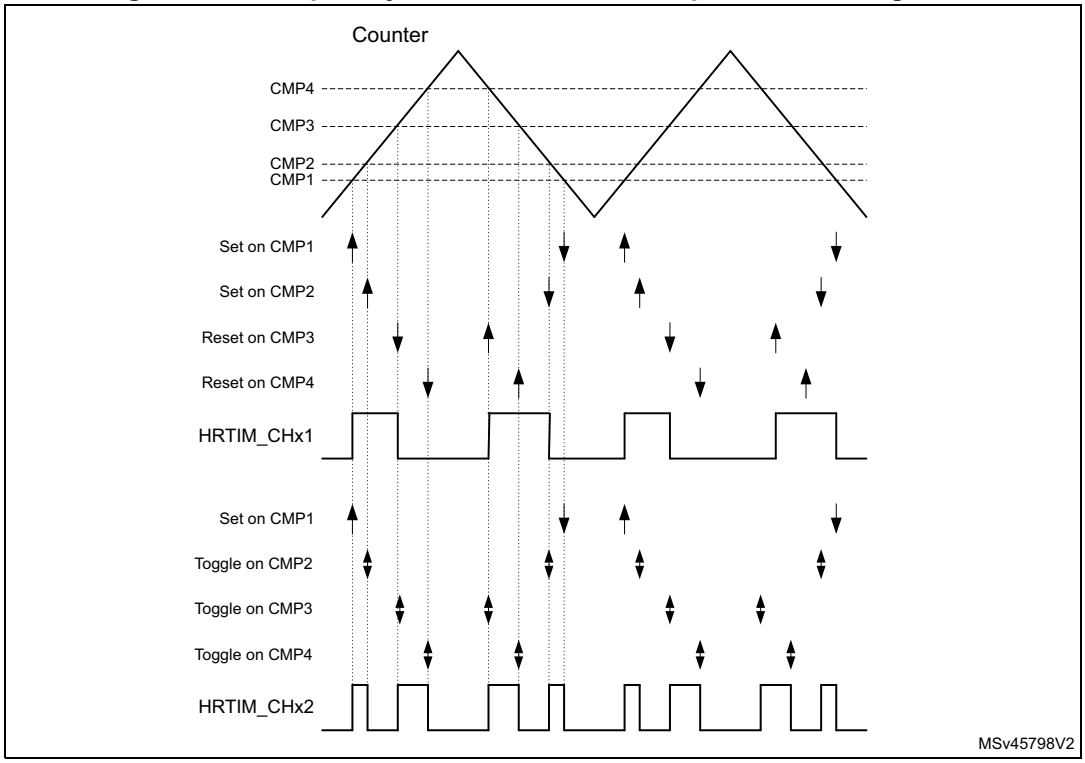
The *Figure 391* below shows how to generate basic waveforms.

Figure 391. Basic symmetric waveform in up-down counting mode



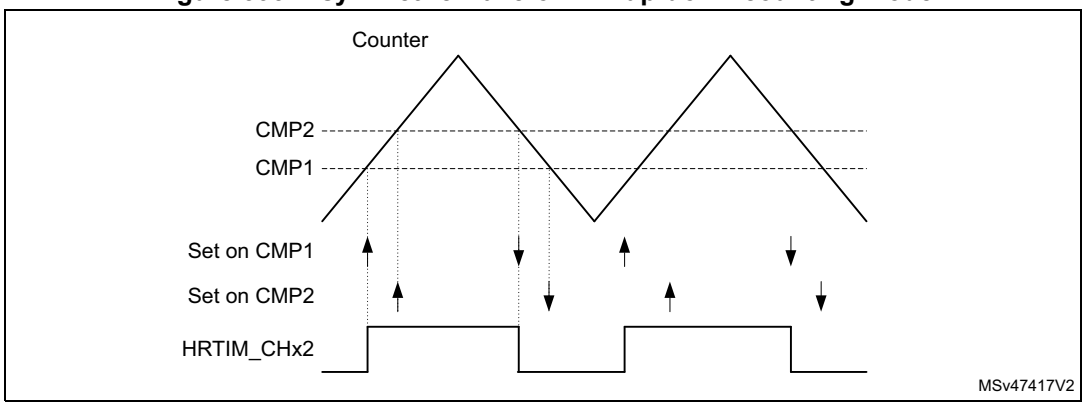
The *Figure 392* below shows how to generate some more complex waveforms, using the 4 available compare units and the toggle mode.

Figure 392. Complex symmetric waveform in up-down counting mode



The *Figure 393* shows how to generate an asymmetric waveform. In this case, it must be noticed that it is necessary to have compare 2 value greater than compare 1 value for the waveform to be asymmetric.

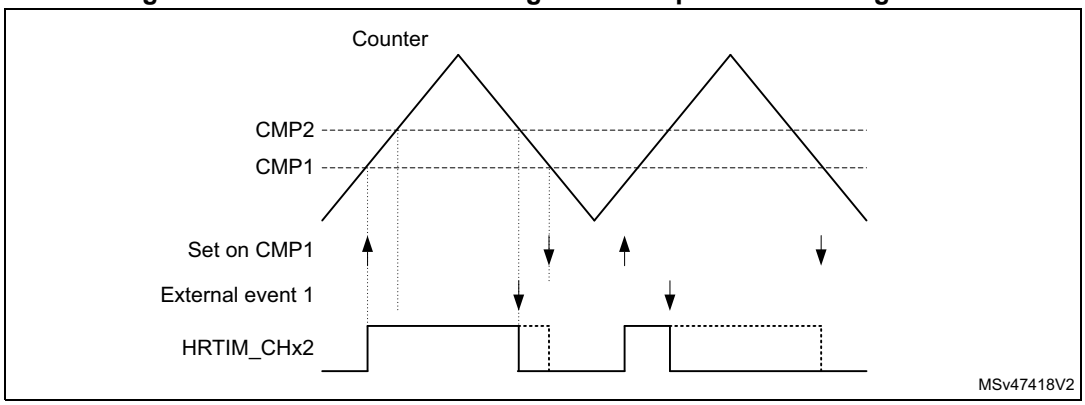
Figure 393. Asymmetric waveform in up-down counting mode



Note: For asymmetric operation, it is required that CMP2 > CMP1.

The behavior of the software forcing bits and external events EXTEVNTA..J is identical for up-only and up-down counting mode. The *Figure 394* below shows how a pulse can be shorten in response to external events.

Figure 394. External event management in up-down counting mode



The up-down counting mode is available for both continuous and single-shot (retriggerable and non-retriggerable) operating modes. A reset causes the counter to re-start from 0. The *Figure 395* below shows the counter behavior in single-shot retriggerable mode, for TIMB.

Figure 395. Interleaved up-down counter operation

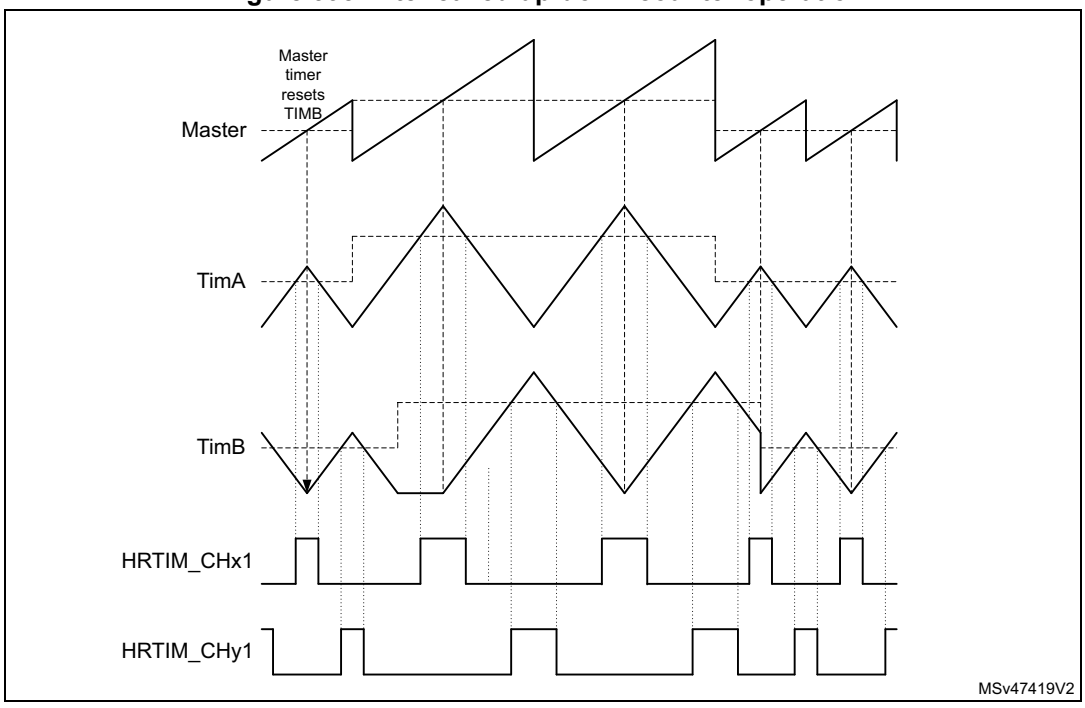
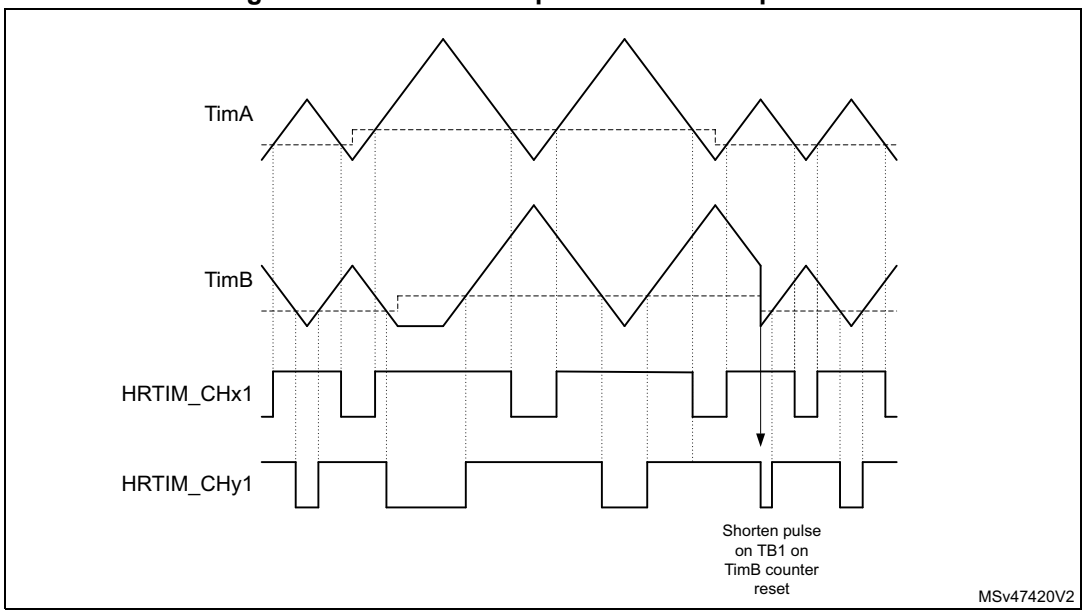


Figure 396. Interleaved up-down counter operation



Note: In up-down counting mode, the compare values must be 3 periods of the f_{HRTIM} clock below the period value (TIMx_PER - 0x60 if CKPSC[2:0] = 0, TIMx_PER - 0x30 if CKPSC[2:0] = 1, TIMx_PER - 0x18 if CKPSC[2:0] = 2,...). This applies for the compare events generated inside the timing unit. For compare events generated in other timing units, it is mandatory to avoid any event occurring within less than 1 period of the f_{HRTIM} clock of a counter direction change (counter at 0, period event or counter reset).

The following features are supported in up-down counting mode:

- Half mode
- Deadtime insertion
- Push-pull mode, alternance push-pull done on when counter = 0 (see *Figure 397*).
- Delayed Idle
- Burst mode
- PWM mode with “greater than” comparison (see *Figure 398*).

Figure 397. Push-pull up-down mode example

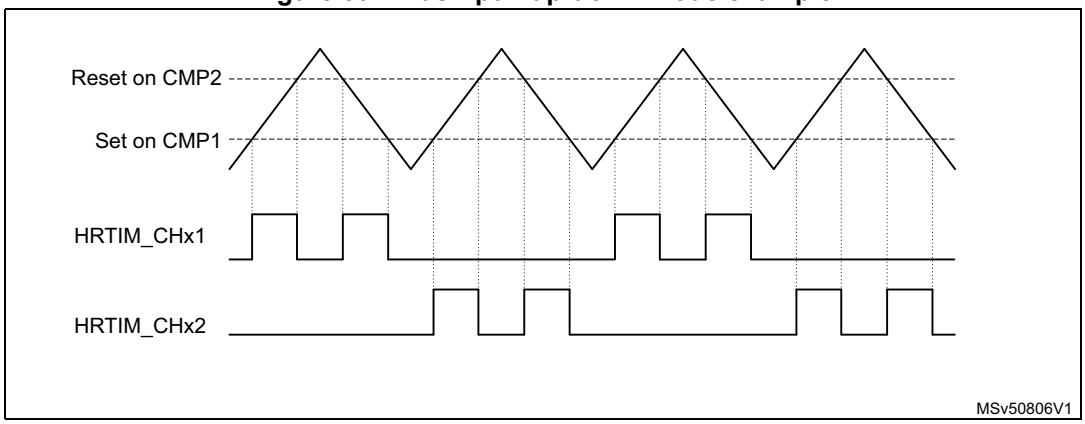
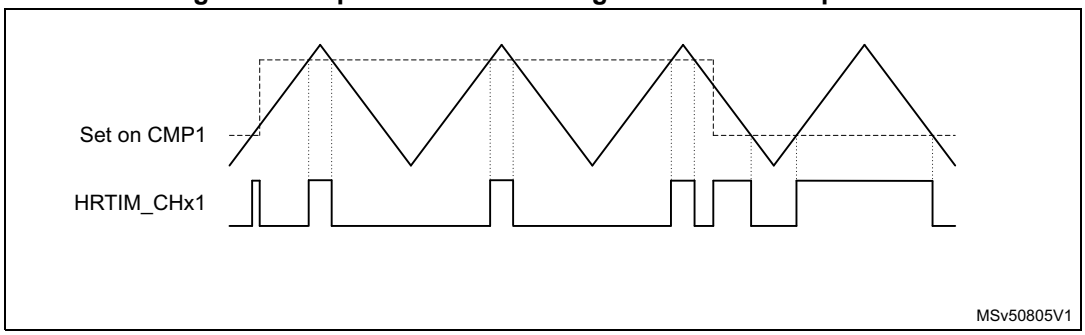


Figure 398. Up-down mode with “greater than” comparison



Caution: The following features are not supported in up-down counting mode:

- Auto-delayed mode
- Balanced Idle
- Triggered-half mode

The capture function is supported with the following differences:

- The capture value is referred to counting start, when up-counting
- The capture value is referred to the PER event when down counting
- The bit 16 of the capture register holds the counting direction status.

The counter roll-over event is defined differently in-up-down counting mode to support various operating condition. It can be either generated:

- When the counter is equal to 0 (“valley” mode)
- When the counter reaches the period value set in the HRTIM_PERxR (“crest” mode)
- When both conditions are met (either 0 or HRTIM_PERxR value).

This event is used for multiple purposes in the HRTIM. The generation mode (valley, crest or both) can be programmed individually depending on the destination. *Table 426* summarizes the use cases and associated roll-over mode (xxROM[1:0]) programming bits in the HRTIM_TIMxCR2 register.

Table 426. Roll-over event destination and mode programming

Roll-over event use	Programming bits
Output set/reset	OUTROM[1:0]
Register content update trigger (transfer from preload to active)	ROM[1:0]
IRQ and/or DMA request trigger	ROM[1:0]
Burst mode clock source and /or burst start trigger	BMROM[1:0]
ADC trigger (see *Section : ADC postscaler* for details)	ADROM[1:0]
External event filtering	ROM[1:0]
Repetition counter decrement	ROM[1:0]
Fault and event counter	FEROM[1:0]

Note: For events where both reset and roll-over are considered (IRQ/DMA and TxRSTU), the xxROM[1:0] influences both the reset and roll-over generation. The reset event is propagated only when xxROM[1:0] = 00 or 01, regardless of the counting direction.

The roll-over event generation is defined as per following xxROM[1:0] bitfield setting:

- xxROM[1:0] = 00: event generated when any of these three conditions are met: counter = 0, counter = HRTIM_PERxR or counter reset.
- xxROM[1:0] = 01: event generated when the counter is equal to 0 (“valley” mode) or when the counter is reset
- xxROM[1:0] = 10: event generated when the counter reaches the period value set in the HRTIM_PERxR (“crest” mode)

The following events are conditioned by the xxROM[1:0] bit fields settings:

- Period event

  It can trigger output set/reset (as per OUTROM[1:0] setting), ADC conversions (as per ADROM[1:0] settings) and burst mode (as per BMROM[1:0] settings), both in continuous and single mode, when the period is explicitly listed as trigger source (for instance ADC1TAPER bit set in HRTIM_ADC1R). When both reset and roll-over events are enabled (for instance ADC1TARST bit set in HRTIM_ADC1R), the period event is considered only in continuous mode (roll-over event). In single-shot mode, no event is generated.
- Roll-over event

  This is similar to period event, for what about xxxROM[1:0] programming, and only valid in continuous mode. In single-shot mode, no event is generated.
- Compare event

  It triggers ADC conversion (as per ADROM[1:0] setting):
  - During up-counting if ADROM = 10
  - During down-counting if ADROM = 01
  - During both up and down-counting if ADROM = 00

Figure 399 here-below shows a push-pull up-down mode with set on period event and OUTROM[1:0] =10.

Figure 399. Up-down mode with output set on period event, for OUTROM[1:0]=10

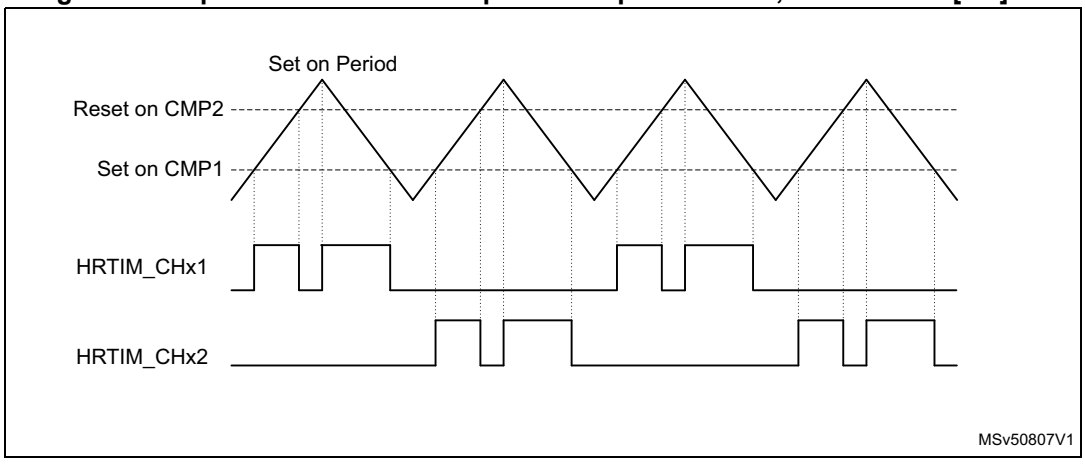
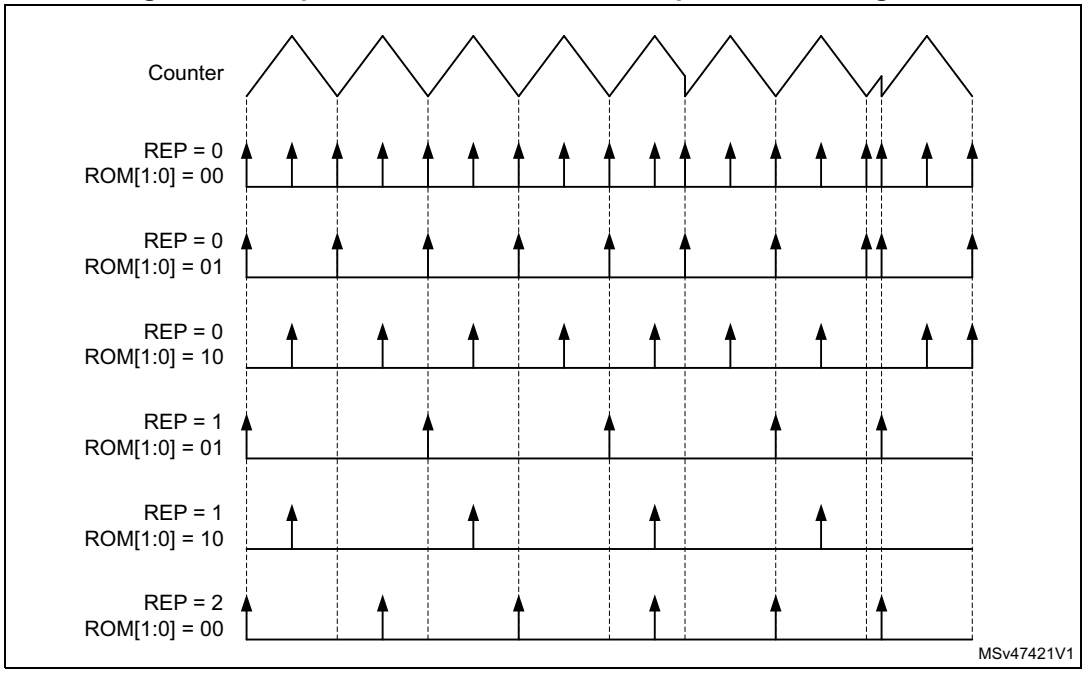


Figure 400 below shows how the repetition counter is decremented in up-down counting mode.

Figure 400. Repetition counter behavior in up-down counting mode



The dual channel DAC triggers are working as for the up-counting mode.

The event blanking and windowing differs so as to have the blanking or windowing done within the output pulse, at a programmable time. The EExFLTR[3:0] codes are depending on the UDM bit setting, as per the *Table 427* below. Whenever the roll-over event is used for blanking or windowing, the ROM[1:0] programming applies for defining when it is generated.

Table 427. EExFLTR[4:0] codes depending on UDM bit setting

EExFLTR[4:0]	Up-counting mode (UDM = 0)	Up/down-counting mode (UDM = 1)
00010	Blanking from counter reset/roll-over to compare 2	Blanking from compare 1 to compare 2, only during the up-counting phase
00100	Blanking from counter reset/roll-over to compare 4	Blanking from compare 3 to compare 4, only during the up-counting phase
01101	Windowing from counter reset/roll-over to compare 2	Windowing from compare 2 to compare 3, only during the up-counting phase
01110	Windowing from counter reset/roll-over to compare 3	Windowing from compare 2 to compare 3, only during the down-counting phase
01111	Windowing from another timing unit: TIMWIN source (see *Table 432* for details)	Windowing from compare 2 during the up-counting phase to compare 3 during the down-counting phase

40.3.7 Set/reset events priorities and narrow pulses management

This section describes how the output waveform is generated when several set and/or reset requests are occurring within 3 consecutive t_{HRTIM} periods.

Case 1: clock prescaler CKPSC[2:0] < 5

An arbitration is performed during each t_{HRTIM} period, in 3 steps:

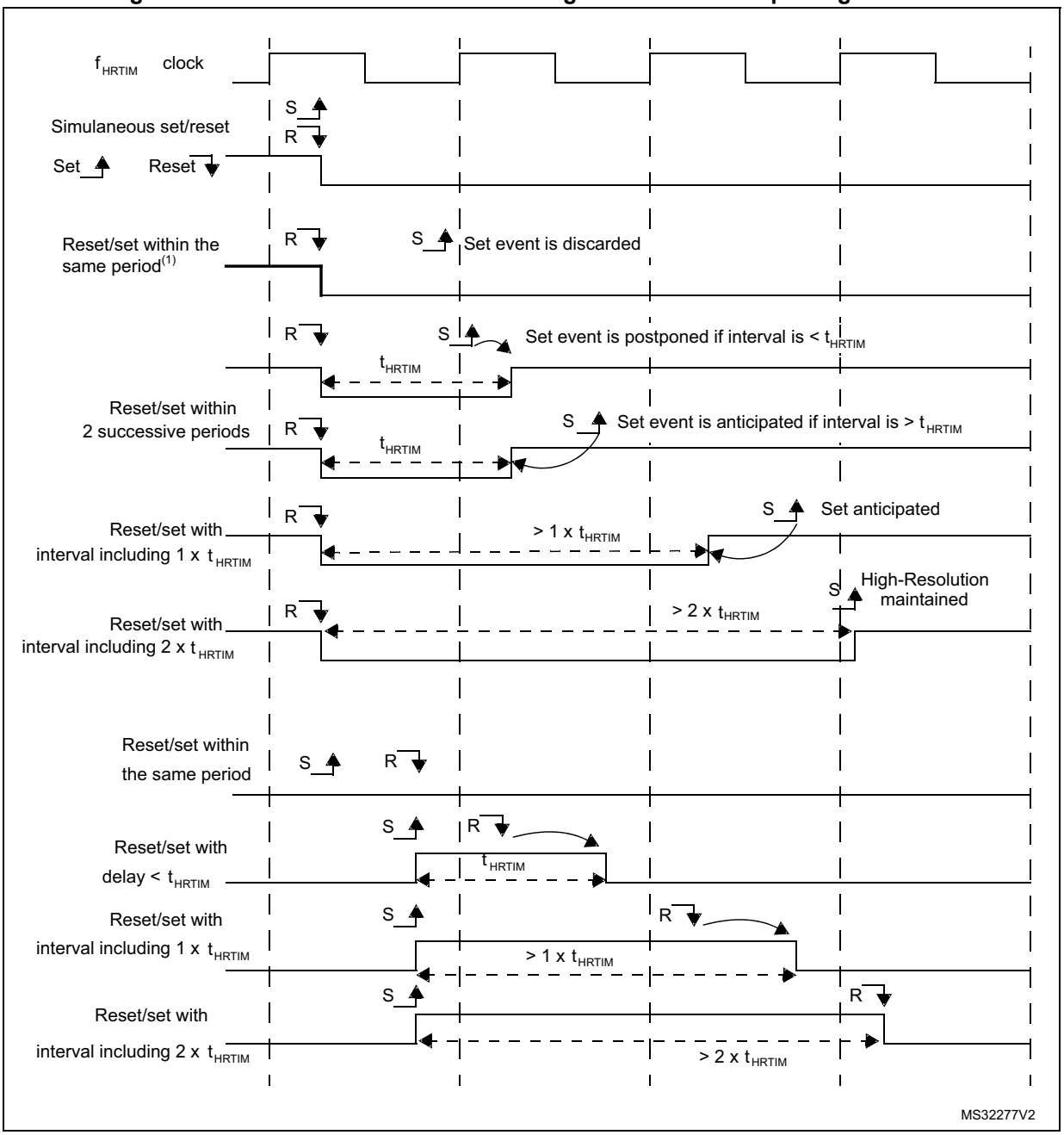
1. For each active event, the desired output transition is determined (set, reset or toggle).
2. A predefined arbitration is performed among the active events, see *Section : Concurrent set requests/Concurrent reset requests*).
3. A high-resolution delay-based arbitration is performed with reset having the highest priority, among the low-resolution events and events having won the predefined arbitration.

When set and reset requests from two different sources occur simultaneously or within the same t_{HRTIM} interval, the reset request has the highest priority, unless the set and reset requests originate from the same timing unit. In this case, the CMPx priority scheme applies. For example, consider CMP2 and CMP4:

- If CMP2 performs a set operation and CMP4 performs a reset operation, the output is reset.
- If CMP2 performs a reset operation and CMP4 performs a set operation, the output is set.

If the interval between set and reset requests is below 2 f_{HRTIM} period, the behavior depends on the time interval and on the alignment with the f_{HRTIM} clock, as shown on *Figure 401*.

Figure 401. Short distance set/reset management for narrow pulse generation



1. If reset and set are generated within the same timing unit, the CMPx priority scheme applies. For example, reset on CMP4 and set on CMP2.

If the set and reset events are generated within the same tHRTIM period, the reset event has the highest priority and the set event is ignored, unless the set and reset requests come from the same timing unit. In this case, the CMPx priority scheme applies.

If the set and reset events are generated with an interval below t_{HRTIM} period, across 2 periods, a pulse of 1 t_{HRTIM} period is generated.

If the set and reset events are generated with an interval below 2 t_{HRTIM} periods, a pulse of 2 t_{HRTIM} periods is generated.

If the set and reset events are generated with an interval between 2 and 3 t_{HRTIM} periods, the high-resolution is available if the interval is over 2 complete t_{HRTIM} periods.

If the set and reset events are generated with an interval above 3 t_{HRTIM} periods, the high-resolution is always available.

Concurrent set requests/Concurrent reset requests

When multiple sources are selected for a set event, an arbitration is performed when the set requests occur within the same f_{HRTIM} clock period.

In case of multiple requests from adjacent timers (TIMEVNT1..9), the request which occurs first is taken into account. The arbitration is done in 2 steps, depending on:

1. the source (CMP4 ⟶ CMP3 ⟶ CMP2 ⟶ CMP1),
2. the delay.

If multiple requests from the master timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and a single request is taken into account, whatever the effective high-resolution setting (from the highest to the lowest priority):

MSTCMP4 → MSTCMP3 → MSTCMP2 → MSTCMP1 → MSTCMPER

Note: It is advised to avoid generating multiple set (reset) requests from the master timer to a given timer with an interval below 3x t_{HRTIM} to maintain the high-resolution.

When multiple requests internal to the timer occur within the same f_{HRTIM} clock period, a predefined arbitration is applied and the requests are taken with the following priority, whatever the effective timing (from highest to lowest):

CMP4 → CMP3 → CMP2 → CMP1 → PER

Note: Practically, this is of a primary importance when multiple compare events can be simultaneously generated or when using autodelayed compare 2 and compare 4 simultaneously (i.e. when the effective set/reset cannot be determined a priori because it is related to an external event). In this case, the highest priority signal must be affected to the CMP4 event.

Last, the highest priority is given to low-resolution events: EXTEVNT1..10, RESYNC (coming from SYNC event if SYNCRSTx or SYNCSTRTx is set or from a software reset), update and software set (SST). The update event is considered as having the largest delay (0x1F if PSC = 0).

As a summary, in case of a close vicinity (events occurring within the same f_{HRTIM} clock period), the effective set (reset) event is arbitrated between:

- Any TIMEVNT1..9 event
- A single source from the master (as per the fixed arbitration given above)
- A single source from the timer
- The “low-resolution events”.

The same arbitration principle applies to concurrent reset requests. In this case, the reset request has the highest priority, unless the set and reset requests originate from the same timing unit. In this situation, the CMPx priority scheme applies.

Case 2: clock prescaler CKPSC[2:0] ≥ 5

The narrow pulse management is simplified when the high-resolution is not effective.

A set or reset event occurring within the prescaler clock cycle is delayed to the next active edge of the prescaled clock (as for a counter reset), even if the arbitration is still performed every t_{HRTIM} cycle.

If a reset event is followed by a set event within the same prescaler clock cycle, the latest event is considered.

40.3.8 External events global conditioning

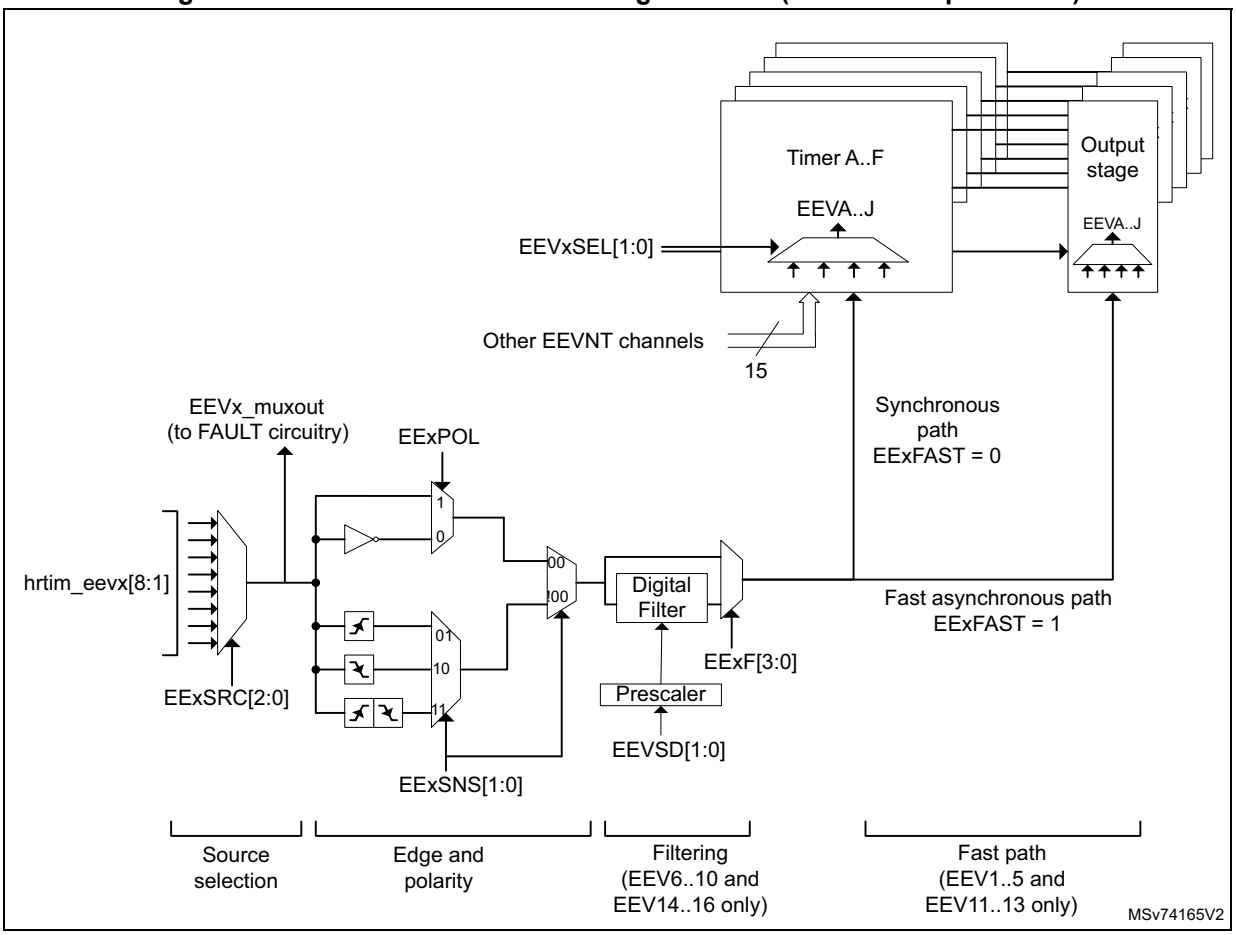
The HRTIM timer can handle events not generated within the timer, referred to as “external event”. These external events come from multiple sources, either on-chip or off-chip:

- built-in comparators,
- digital input pins (typically connected to off-chip comparators and zero-crossing detectors),
- on-chip events for other peripheral (ADC’s analog watchdogs and general purpose timer trigger outputs).

The external events conditioning circuitry allows to select the signal source for a given channel (with a 8:1 multiplexer) and to convert it into an information that can be processed by the crossbar unit (for instance, to have an output reset triggered by a falling edge detection on an external event channel).

Up to 16 external event channels can be conditioned and are available simultaneously for any of the 6 timers. This conditioning is common to all timers, since this is usually dictated by external components (such as a zero-crossing detector) and environmental conditions (typically the filter set-up is related to the applications noise level and signature). *Figure 402* presents an overview of the conditioning logic for a single channel.

Figure 402. External event conditioning overview (1 channel represented)



The 16 external events are initialized using the HRTIM_EECR1, HRTIM EECR2, HRTIM_EECR4, HRTIM_EECR5 and HRTIM EECR7 registers:

- To select up to 8 sources with the EExSRC[2:0] bits,
- To select the sensitivity with EExSNS[1:0] bits, to be either level-sensitive or edge-sensitive (rising, falling or both),
- To select the polarity, in case of a level sensitivity, with EExPOL bit,
- To have a low latency mode, with EExFAST bits (see *Section : Latency to external events*), for external events 1 to 5 and external events 11 to 13.

Note: The external events used as triggers for reset, capture, burst mode, ADC triggers and delayed protection are edge-sensitive even if EESNS bit is reset (level-sensitive selection): if POL = 0 the trigger is active on external event rising edge, while if POL = 1 the trigger is active on external event falling edge.

The external events are discarded as long as the counters are disabled (TxCEN bit reset) to prevent any output state change and counter reset, except if they are used as ADC triggers.

Additionally, it is possible to enable digital noise filters, for external events 6 to 10 using EExF[3:0] bits in the HRTIM_EECR3 register and for external events 14 to 16 using EExF[3:0] bits in the HRTIM_EECR6 register.

A digital filter is made of a counter in which a number N of valid samples is needed to validate a transition on the output. If the input value changes before the counter has

reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered, depending on the sampling clock and on the filter length (number of valid samples expected).

The sampling clock is either the f_{HRTIM} clock or a specific prescaled clock f_{EEVS} derived from f_{HRTIM}, defined with EEVSD[1:0] bits in HRTIM_EECR3 register.

Table 428 summarizes the available features associated with each of the 10 external events channels.The features and sources are summarized in *Table 411*.

Table 428. External events features

External event channel	Fast mode	Digital filter	Balanced fault timer A,B,C	Balanced fault timer D,E,F
hrtim_eev1[8:1]	Yes	-	-	-
hrtim_eev2[8:1]	Yes	-	-	-
hrtim_eev3[8:1]	Yes	-	-	-
hrtim_eev4[8:1]	Yes	-	-	-
hrtim_eev5[8:1]	Yes	-	-	-
hrtim_eev6[8:1]	-	Yes	Yes	-
hrtim_eev7[8:1]	-	Yes	Yes	-
hrtim_eev8[8:1]	-	Yes	-	Yes
hrtim_eev9[8:1]	-	Yes	-	Yes
hrtim_eev10[8:1]	-	Yes	-	-
hrtim_eev11[8:1]	Yes	-	-	-
hrtim_eev12[8:1]	Yes	-	-	-
hrtim_eev13[8:1]	Yes	-	-	-
hrtim_eev14[8:1]	-	Yes	-	-
hrtim_eev15[8:1]	-	Yes	-	-
hrtim_eev16[8:1]	-	Yes	-	-

The *Table 429* provides the mapping between EEVA..EEVJ events and external events hrtim_eev1 to hrtim_eev16. This is configured using the EEVxSEL[1:0] bitfield in the HRTIM_EESELxR register.

Table 429. External events selection

External event channel x = A..J	EEVxSEL[1:0]=00	EEVxSEL[1:0]=01	EEVxSEL[1:0]=10	EEVxSEL[1:0]=11
EEVA	hrtim_eev1	hrtim_eev11	hrtim_eev1 AND hrtim_eev2	hrtim_eev1 AND hrtim_eev11
EEVB	hrtim_eev2	hrtim_eev12	hrtim_ee2 AND hrtim_eev3	hrtim_eev2 AND hrtim_eev12

Table 429. External events selection

External event channel x = A..J	EEVxSEL[1:0]=00	EEVxSEL[1:0]=01	EEVxSEL[1:0]=10	EEVxSEL[1:0]=11
EEVC	hrtim_eev3	hrtim_eev13	hrtim_ee3 AND hrtim_eev4	hrtim_eev3 AND hrtim_eev13
EEVD	hrtim_eev4	hrtim_eev14	hrtim_ee4 AND hrtim_eev5	hrtim_eev4 AND hrtim_eev14
EEVE	hrtim_eev5	hrtim_eev15	hrtim_ee5 AND hrtim_eev6	hrtim_eev5 AND hrtim_eev15
EEVF	hrtim_eev6	hrtim_eev16	hrtim_ee6 AND hrtim_eev7	hrtim_eev6 AND hrtim_eev16
EEVG	hrtim_eev7	hrtim_eev11	hrtim_ee7 AND hrtim_eev8	hrtim_eev7 AND hrtim_eev11
EEVH	hrtim_eev8	hrtim_eev12	hrtim_ee8 AND hrtim_eev9	hrtim_eev8 AND hrtim_eev12
EEVI	hrtim_eev9	hrtim_eev13	hrtim_ee9 AND hrtim_eev10	hrtim_eev9 AND hrtim_eev13
EEVJ	hrtim_eev10	hrtim_eev14	hrtim_ee10 AND hrtim_eev11	hrtim_eev10 AND hrtim_eev14

The external events can be combined (hrtim_eevy AND hrtim_eevz) with EEVxSEL[1:0] = 10 or 11.

They can be combined solely if they both are in same mode:

- hrtim_eey (with fast mode option) and hrtim_eevz (with fast mode option), with both EEyFAST = EEzFAST = 0 or EEyFAST = EEzFAST = 1
- hrtim_eey (with digital filter option) and hrtim_eevz (with fast mode option) with EEzFAST = 0

It is not possible to combine hrtim_eey (with digital filter option) and hrtim_eevz (with fast mode option) with EEzFAST = 1.

When EEVA..EEVJ is resulting from a combination of 2 external events (EEVxSEL[1:0] = 10 or 11), at least one of the 2 inputs must be level sensitive. Having a level sensitive input combined with an edge sensitive input results in an edge sensitive combined signal. This gives the possibility to let an external event be generated depending on the level of an other external event (AND function).

Latency to external events

The external event conditioning gives the possibility to adjust the external event processing time (and associated latency) depending on performance expectations:

- A regular operating mode, in which the external event is resampled with the clock before acting on the output crossbar. This adds some latency but gives access to all

crossbar functionalities. It enables the generation of an externally triggered high-resolution pulse.

- A fast operating mode, in which the latency between the external event and the action on the output is minimized. This mode is convenient for ultra-fast over-current protections, for instance.

EExFAST bits in the HRTIM_EECR1 and HRTIM_EECR4 registers allow to define the operating for channels 1 to 5 and channels 11 to 13. This influences the latency and the jitter present on the output pulses, as summarized in the table below.

Table 430. Output set/reset latency and jitter versus external event operating mode

EExFAST	Response time latency	Response time jitter	Jitter on output pulse (counter reset by ext. event)
0	5 to 6 cycles of f_{HRTIM} clock	1 cycles of f_{HRTIM} clock	No jitter, pulse width maintained with high-resolution
1	Minimal latency (depends whether the comparator or digital input is used)	Minimal jitter	1 cycle of f_{HRTIM} clock jitter pulse width resolution down to t_{HRTIM}

The EExFAST mode is only available with level-sensitive programming (EExSNS[1:0] = 00); the edge-sensitivity cannot be programmed.

It is possible to apply event filtering to fast external events (both blanking and windowing with EExFLTR[3:0] ≠ 0000, see *Section 40.3.9*). In this case, EExLTCHx bit must be reset: the postponed mode is not supported, neither the windowing timeout feature.

Note: The external event configuration (source and polarity) must not be modified once the related EExFAST bit is set.

A fast external event cannot be used to toggle an output: if must be enabled either in HRTIM_SETxyR or HRTIM_RSTxyR registers, not in both.

When a set and a reset event - from 2 independent fast external events - occur simultaneously, the reset has the highest priority in the crossbar and the output becomes inactive.

When EExFAST bit is set, the output cannot be changed during the 11 f_{HRTIM} clock periods following the external event.

Figure 403 and *Figure 404* give practical examples of the reaction time to external events, for output set/reset and counter reset.

Figure 403. Latency to external events (counter reset and output set)

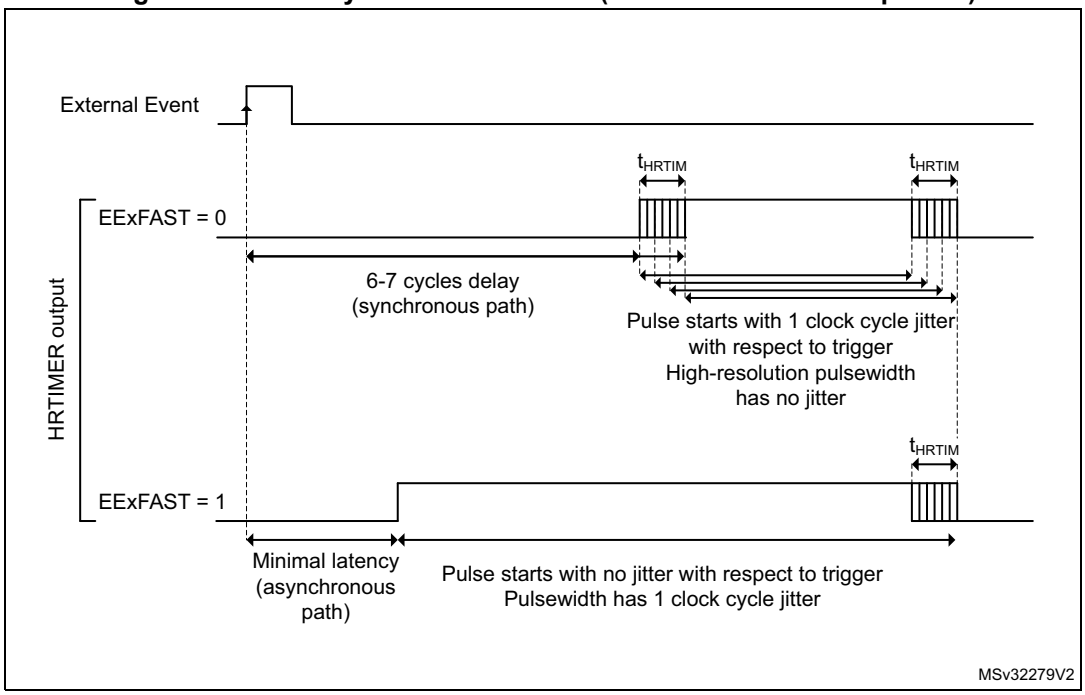
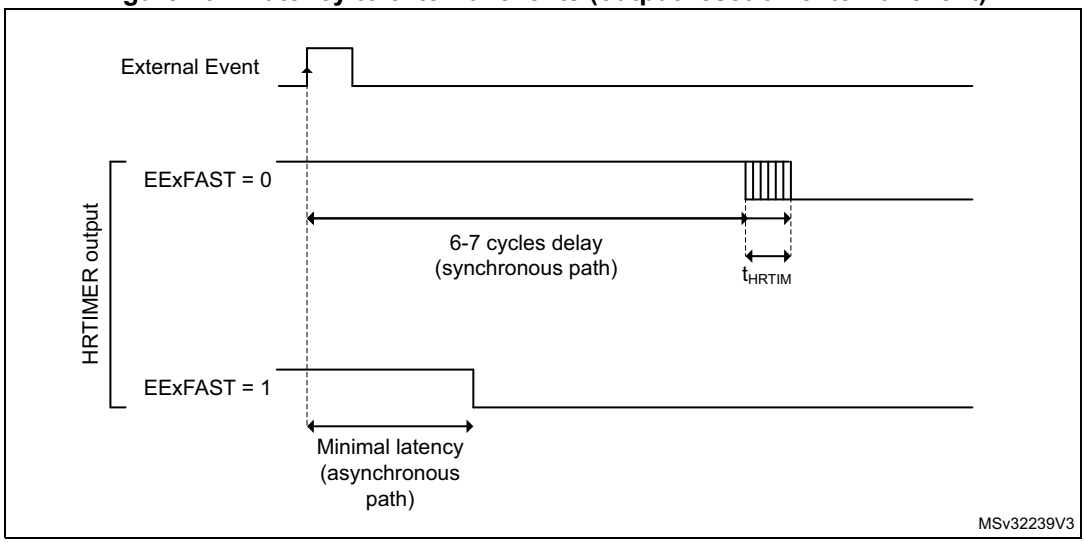


Figure 404. Latency to external events (output reset on external event)



40.3.9 External event filtering in timing units

Once conditioned, the 10 external events are available for all timing units.

They can be used directly and are active as soon as the timing unit counter is enabled (TxCEN bit set).

They can also be filtered to have an action limited in time, usually related to the counting period. Two operations can be performed:

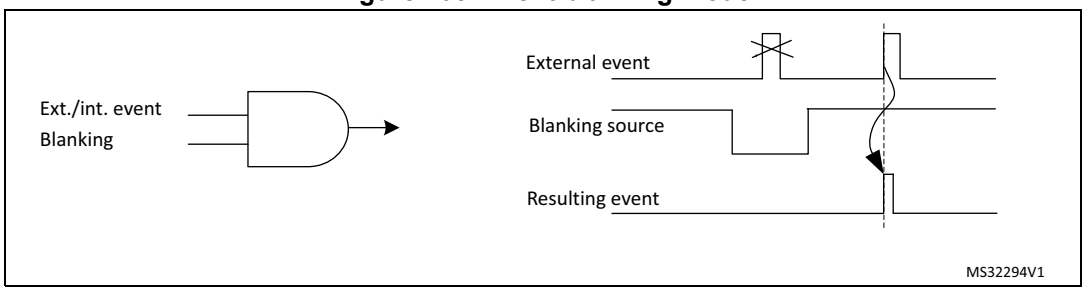
- Blanking, to mask external events during a defined time period,
- Windowing, to enable external events only during a defined time period.

These modes are enabled using HRTIM_EExFLTR[4:0] bits in the HRTIM_EEFxR1 and HRTIM_EEFxR2 registers. Each of the 6 timer A..F timing units has its own programmable filter settings for the 10 external events.

Blanking mode

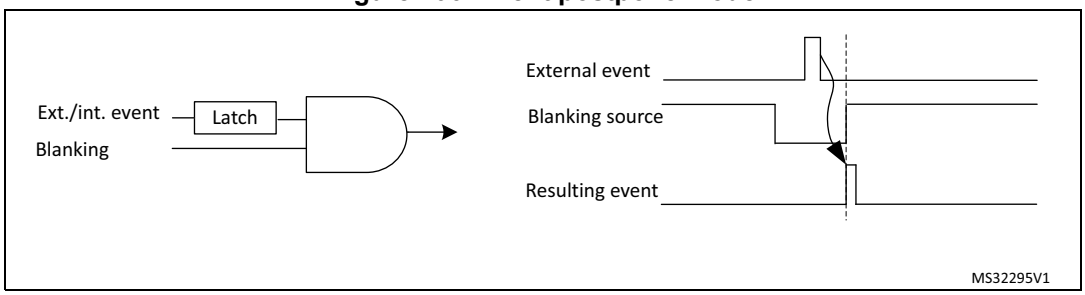
In event blanking mode (see *Figure 405*), the external event is ignored if it happens during a given blanking period. This is convenient, for instance, to avoid a current limit to trip on switching noise at the beginning of a PWM period. This mode is active for EExFLTR[4:0] bitfield values ranging from 00001 to 01100.

Figure 405. Event blanking mode



In event postpone mode, the external event is not taken into account immediately but is memorized (latched) and generated as soon as the blanking period is completed, as shown on *Figure 406*. This mode is enabled by setting EExLTCH bit in HRTIM_EEFxR1 and HRTIM_EEFxR2 registers.

Figure 406. Event postpone mode



The blanking signal comes from several sources:

- the timer itself: the blanking lasts from the counter reset to the compare match (EExFLTR[4:0] = 00001 to 00100 for compare 1 to compare 4). In up/down mode (UDM bit set to 1), the counter reset event is defined as per the ROM[1:0] bit setting.
- from other timing units (EExFLTR[4:0] = 00101 to 01100): the blanking lasts from the selected timing unit counter reset to one of its compare match, or can be fully programmed as a waveform on Tx2 output. In this case, events are masked as long as the Tx2 signal is inactive (it is not necessary to have the output enabled, the signal is taken prior to the output stage).

The EExFLTR[4:0] configurations from 00101 to 01100 are referred to as TIMFLTR1 to TIMFLTR8 in the bit description, and differ from one timing unit to the other. *Table 431* gives the 8 available options per timer: CMPx refers to blanking from counter reset to compare match, Tx2 refers to the timing unit TIMx output 2 waveform defined with HRTIM_SETx2 and HRTIM_RSTx2 registers. For instance, timer B (TIMFLTR6) is timer C output 2 waveform.

Table 431. Filtering signals mapping per timer

Source		Timer A				Timer B				Timer C				Timer D				Timer E				Timer F			
		CMP1	CMP2	CMP4	TA2	CMP1	CMP2	CMP4	TB2	CMP1	CMP2	CMP4	TC2	CMP1	CMP2	CMP4	TD2	CMP1	CMP2	CMP4	TE2	CMP1	CMP2	CMP4	TF2
Destination	Timer A	-				1	-	2	3	4	-	5	-	7	-	-	-	-	8	-	-	6	-	-	-
	Timer B	1	-	2	3	-				4	5	-	-	-	7	-	-	8	-	-	-	-	6	-	-
	Timer C	-	1	-	-	2	-	3	-	-				5	-	6	7	-	-	8	-	4	-	-	-
	Timer D	1	-	-	-	-	2	-	-	3	4	-	5	-				6	-	7	-	-	-	8	-
	Timer E	-	1	-	-	2	-	-	-	3	-	-	-	6	-	7	8	-				-	-	4	5
	Timer F	-	-	1	-	-	2	-	-	-	-	3	-	-	4	5	-	6	-	7	8	-			

Figure 407 and *Figure 408* give an example of external event blanking for all edge and level sensitivities, in regular and postponed modes.

Figure 407. External event blanking with edge-sensitive trigger

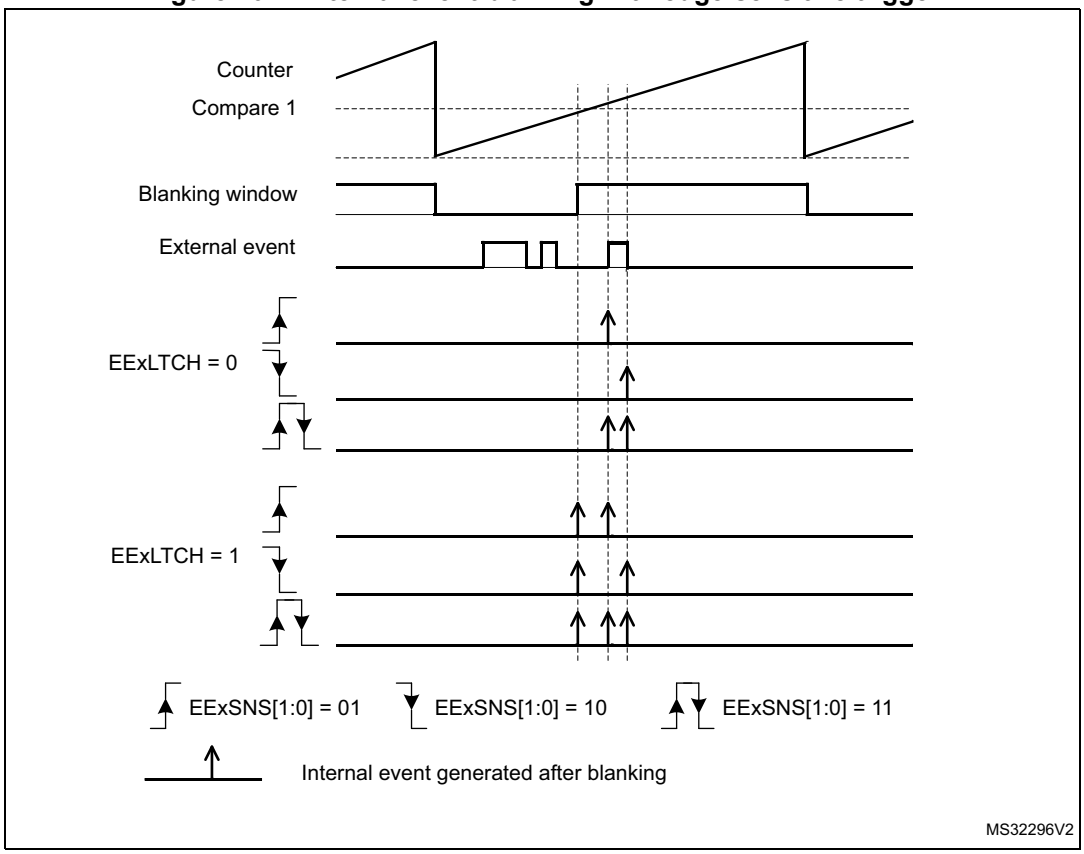
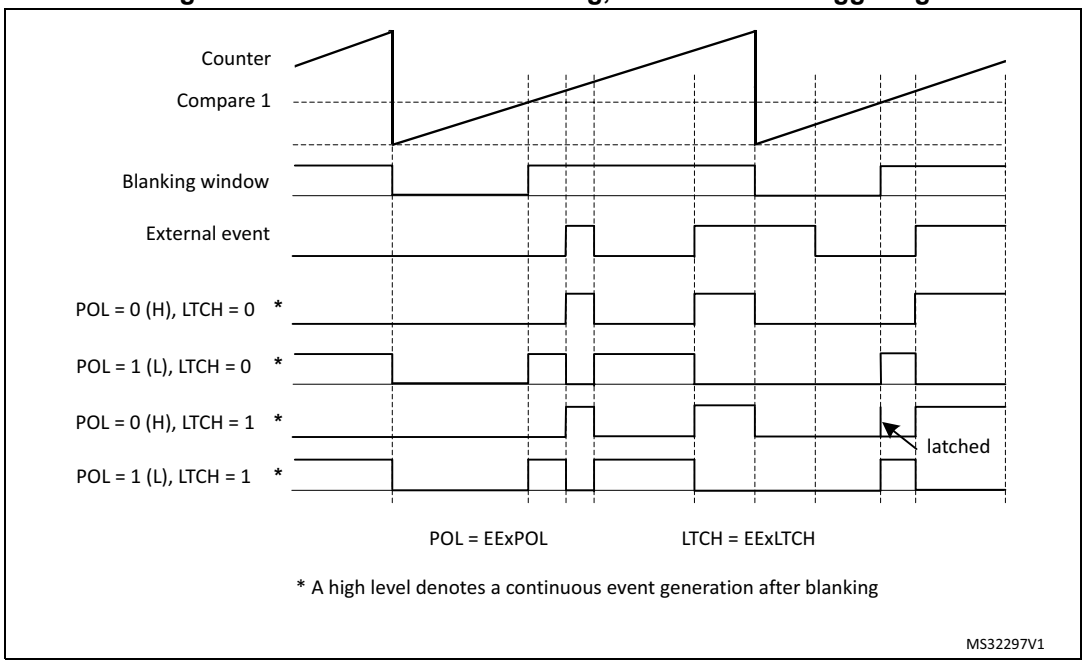


Figure 408. External event blanking, level sensitive triggering



Push-pull blanking mode

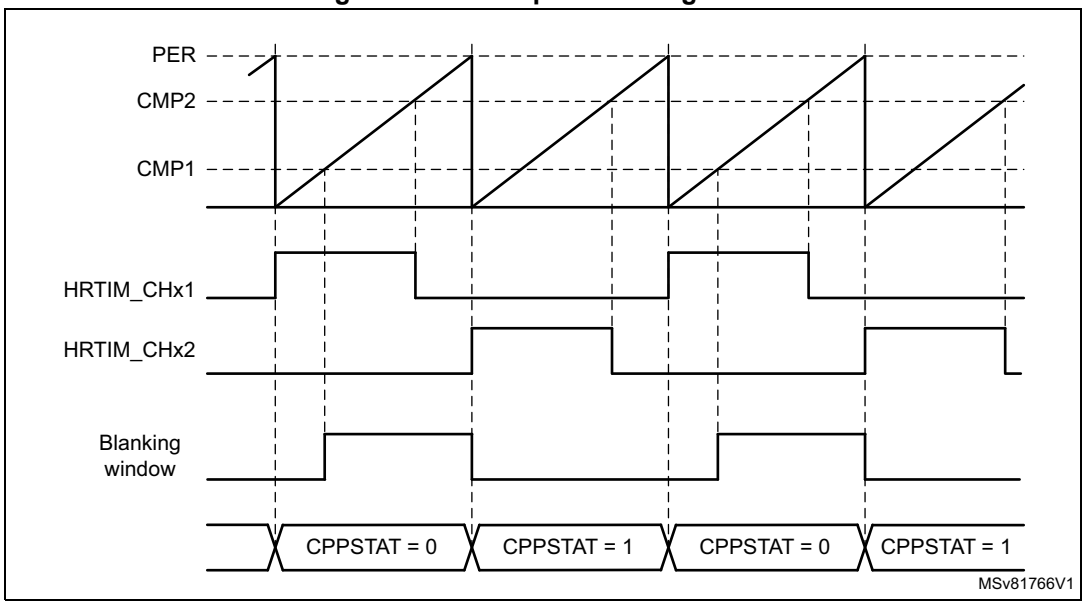
A push-pull specific blanking mode differentiates blanking from one period to the other, as shown in *Figure 409*. Select this mode by programming EExFLTR[4:0] = 11100.

It works as follows:

- During the push phase (CPPSTAT = 0), blanking lasts from the reset/roll-over event to the CMP1 match.
- During the pull phase (CPPSAT = 1), the blanking is always active.

This mode ensures that an external event that generates a balanced idle mode is considered only during the push phase and is rejected if it occurs during the pull phase.

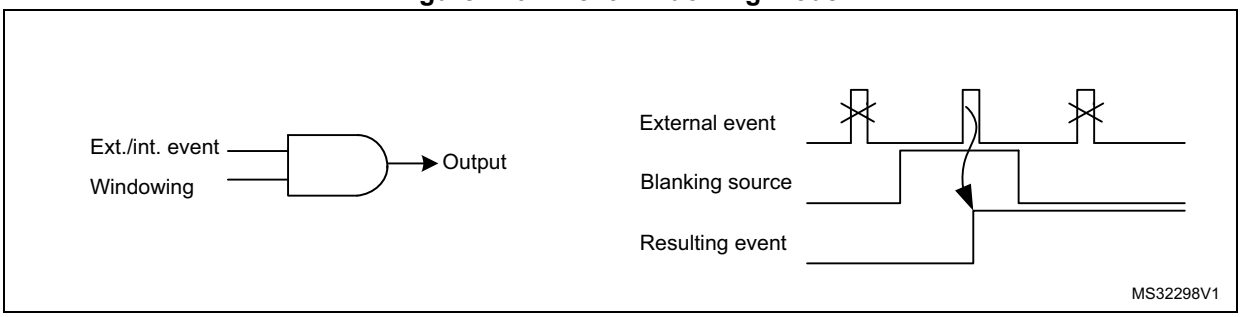
Figure 409. Push-pull blanking mode



Windowing mode

In event windowing mode, the event is taken into account only if it occurs within a given time window, otherwise it is ignored. This mode is active for EExFLTR[4:0] ranging from 01101 to 01111.

Figure 410. Event windowing mode



EExLTCH bit in EEFxR1 and EEFxR2 registers allows to latch the signal, if set to 1: in this case, an event is accepted if it occurs during the window but is delayed at the end of it.

- If EExLTCH bit is reset and the signal occurs during the window, it is passed through directly.
- If EExLTCH bit is reset and no signal occurs, a timeout event is generated at the end of the window.

A use case of the windowing mode is to filter synchronization signals. The timeout generation allows to force a default synchronization event, when the expected synchronization event is lacking (for instance during a converter start-up).

There are 3 sources for each external event windowing, coded as follows:

- 01101 and 01110: the windowing lasts from the counter reset to the compare match (respectively compare 2 and compare 3). In up/down mode (UDM bit set to 1), the counter reset event is defined as per the ROM[1:0] bit setting.
- 01111: the windowing is related to another timing unit and lasts from its counter reset to its compare 2 match. The source is described as TIMWIN in the bit description and is given in *Table 432*. As an example, the external events in timer B can be filtered by a window starting from timer A counter reset to timer A compare 2.

Table 432. Windowing signals mapping per timer (EExFLTR[4:0] = 01111)

Destination	Timer A	Timer B	Timer C	Timer D	Timer E	Timer F
TIMWIN (source)	Timer B CMP2	Timer A CMP2	Timer D CMP2	Timer C CMP2	Timer F CMP2	Timer E CMP2
TIMWIN2 (source)	Timer B CMP2 to reset/roll-over	Timer A CMP2 to reset/roll-over	Timer D CMP2 to reset/roll-over	Timer C CMP2 to reset/roll-over	Timer F CMP2 to reset/roll-over	Timer E CMP2 to reset/roll-over
TIMWIN3 (source)	Timer B CMP1 to CMP2	Timer A CMP1 to CMP2	Timer D CMP1 to CMP2	Timer C CMP1 to CMP2	Timer F CMP1 to CMP2	Timer E CMP1 to CMP2

Note: The timeout event generation is not supported if the external event is programmed in fast mode.

Figure 411 and *Figure 412* present how the events are generated for the various edge and level sensitivities, as well as depending on EExLTCH bit setting. Timeout events are specifically mentioned for clarity reasons.

Figure 411. External event windowing with edge-sensitive trigger

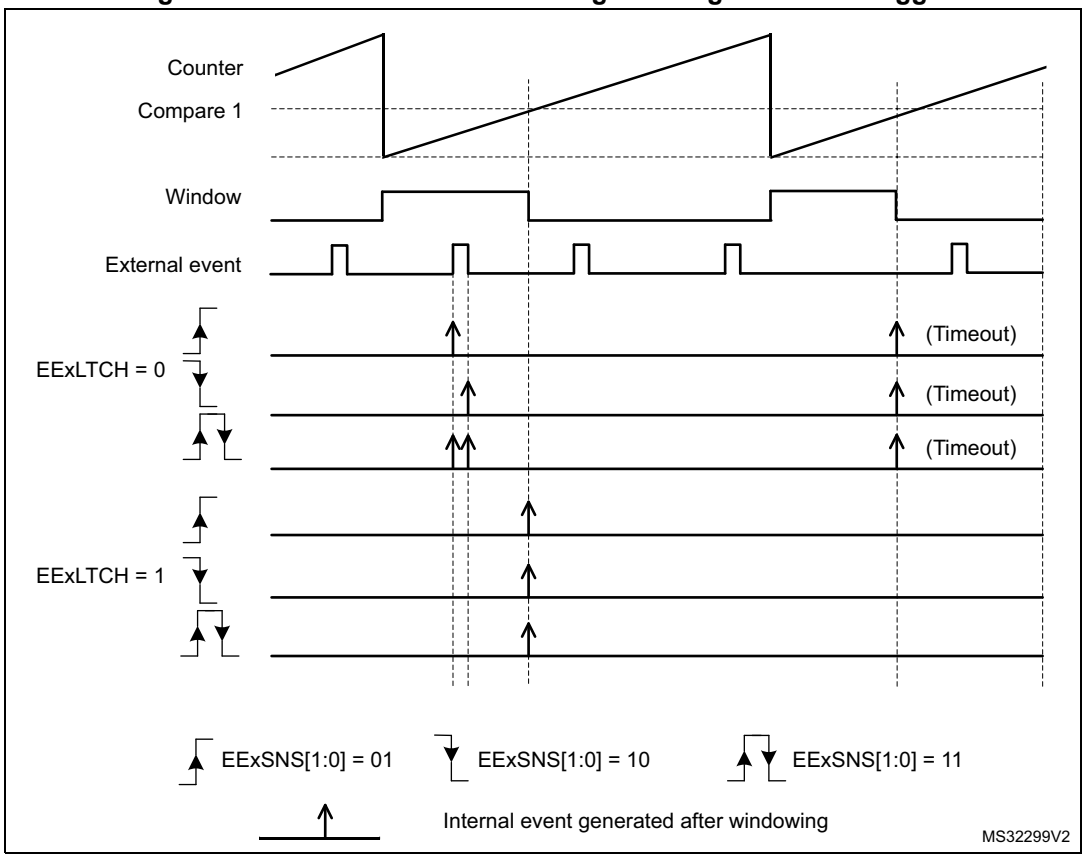
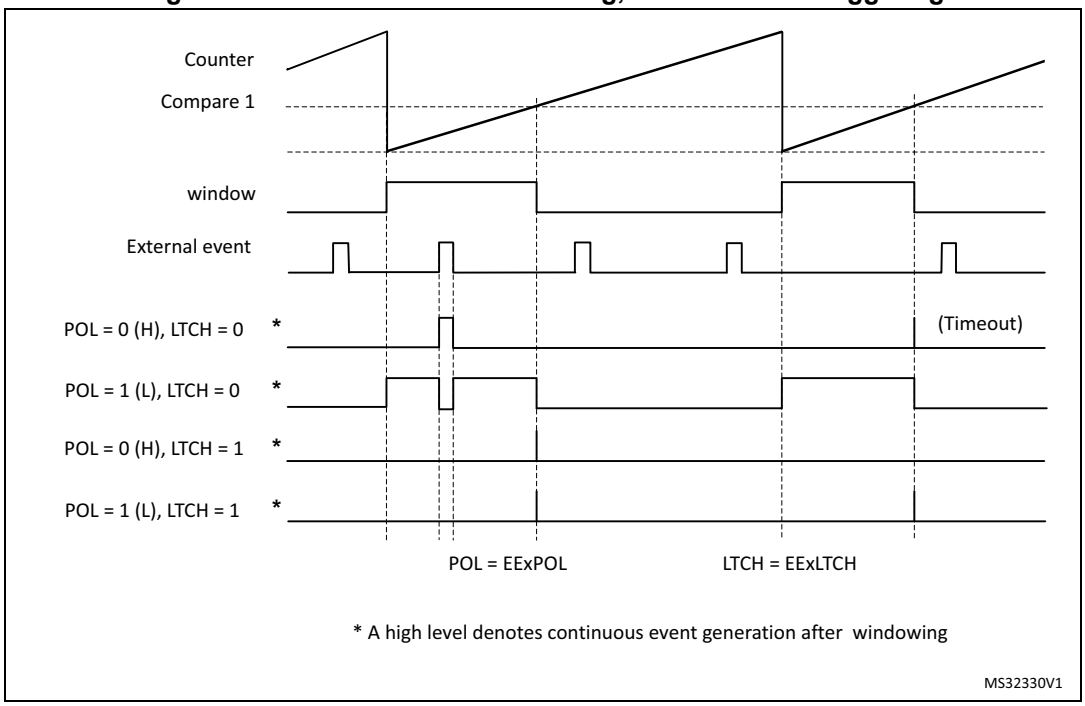


Figure 412. External event windowing, level sensitive triggering

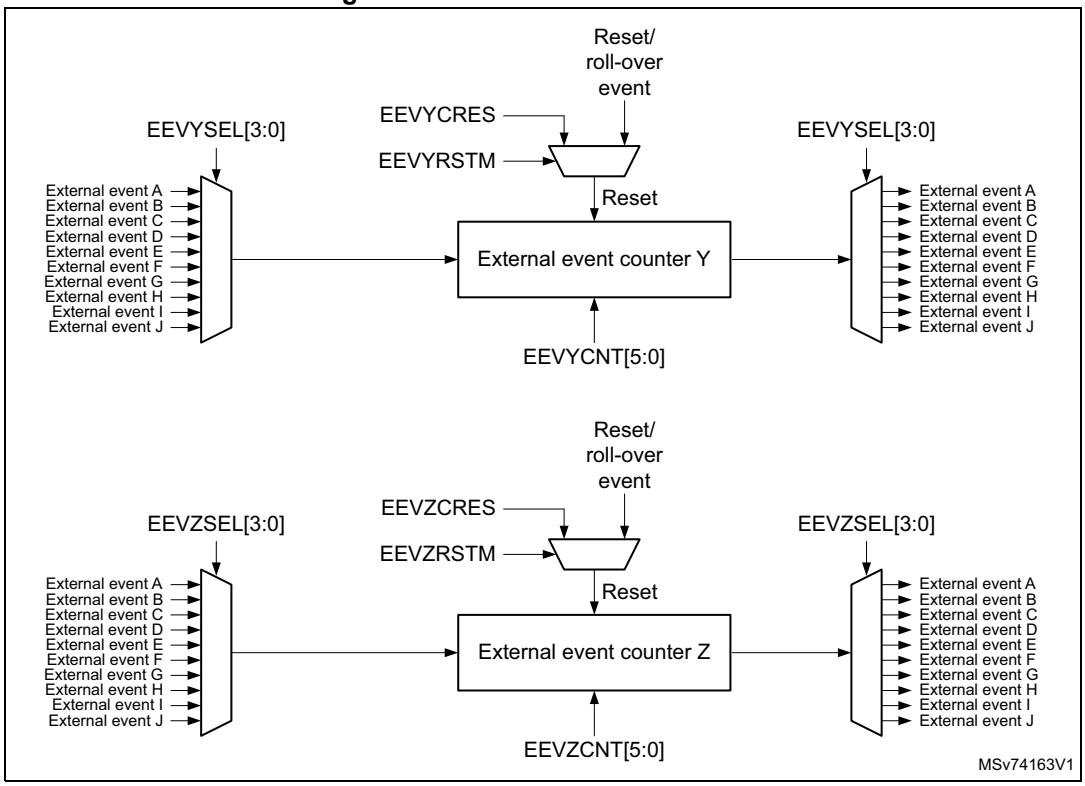


External event counter

Each timing unit also features two external event counters following the filtering unit, typically for valley skipping implementation.

The circuitry allows to filter any of the 10 external events filtered, as shown on *Figure 413*.

Figure 413. External event counters



The counters Y and Z are enabled using the EEVxCE bit in the HRTIM_EEFxR3 register. This mode is only valid for edge-sensitive external events (EExSNS[1:0] bitfield = 01, 10 or 11), when EEVA..EEVJ selection is a single external event EEV1..16 (EEVxSEL[1:0] = 00 or 01).

When EEVA..EEVJ is resulting from a combination of 2 external events (EEVxSEL[1:0] = 10 or 11), at least one of the 2 inputs must be level sensitive. Having a level sensitive input combined with an edge sensitive input results in an edge sensitive combined signal. This gives the possibility to let an external event be generated depending on the level of an other external event (AND function).

The external event is propagated to the timer only if the number of active edges is greater or equal to the value programmed in (EEVxCNT[5:0]+1).

Two operating modes are available:

- When the EEVxRSTM bit is reset, the external event counter is reset on each reset/roll-over event: the external event is active only if it appears several times within a given PWM period
- When the EEVxRSTM bit is set, the external event counter is reset only if the event did not on appear during the last PWM period. This a cumulative mode, where the event must occur at least once during multiple PWM period, as shown on *Figure 414* below.

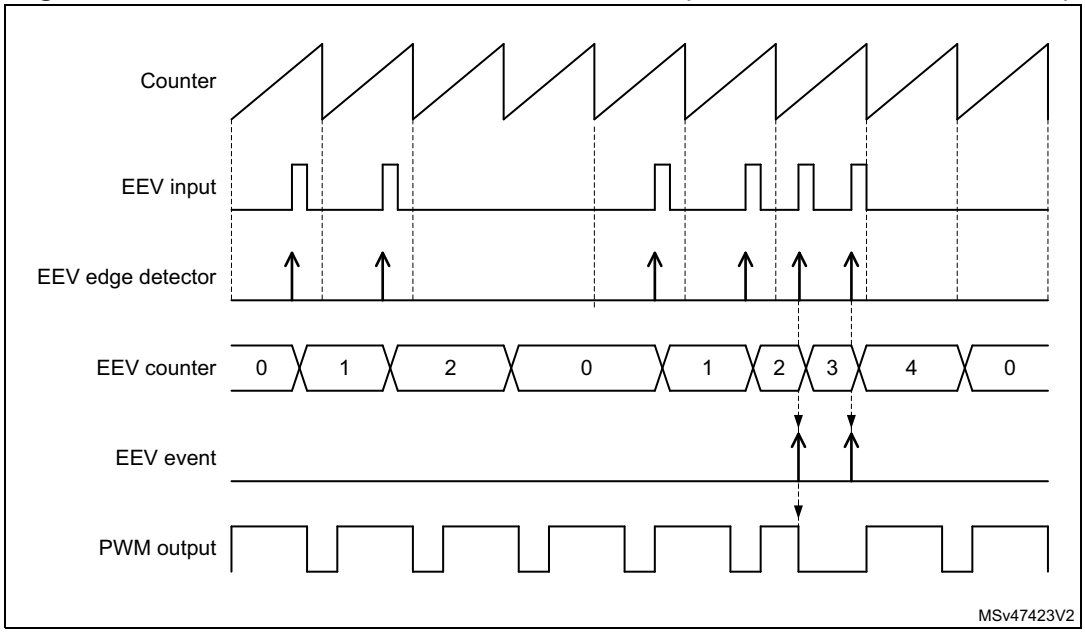
Note: If the 2 external event counters are operating on the same external event (EEVYSEL[3:0] equal to EEVZSEL[3:0]) the external counter Y only is considered.

The external event counter must be enabled after having programmed the counter value (the EEVxCE bit must be set after having written the EEVxCNT[5:0] bits).

Once the counter is enabled, the EEVxCNT[5:0] bits can then be changed on-the-fly at any time. The new value is taken into account on the following reset/rollover event as per the EEVxRSTM bit programming, or after a software reset (EEVxCRES bit set).

The EEVxSEL[3:0] bitfield must not be modified once the EEVxCE bit is set.

Figure 414. External event counter cumulative mode (EEVxRSTM = 1, EEVxCNT = 2)



40.3.10 Delayed protection

The HRTIM features specific protection schemes, typically for resonant converters when it is necessary to shut down the PWM outputs in a delayed manner, either once the active pulse is completed or once a push-pull period is completed. These features are enabled with DLYPRTEN bit in the HRTIM_OUTxR register, and are using specific external event channels.

Delayed idle

In this mode, the active pulse is completed before the protection is activated. The selected external event causes the output to enter in idle mode at the end of the active pulse (defined by an output reset event in HRTIM_RSTx1R or HRTIM_RSTx2R).

Once the protection is triggered, the idle mode is permanently maintained but the counter continues to run, until the output is re-enabled. Tx1OEN and Tx2OEN bits are not affected by the delayed idle entry. To exit from delayed idle and resume operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1. The output state changes on the first transition to an active state following the output enable command.

Note: The delayed idle mode cannot be exited immediately after having been entered, before the active pulse is completed: it is mandatory to make sure that the outputs are in idle state

before resuming the run mode. This can be done by waiting up to the next period, for instance, or by polling the O1CPY and/or O2CPY status bits in the TIMxISR register.

The delayed idle mode can be applied to a single output (DLYPRT[2:0] = x00 or x01) or to both outputs (DLYPRT[2:0] = x10).

An interrupt or a DMA request can be generated in response to a Delayed Idle mode entry. The DLYPRT flag in HRTIM_TIMxISR is set as soon as the external event arrives, independently from the end of the active pulse on output.

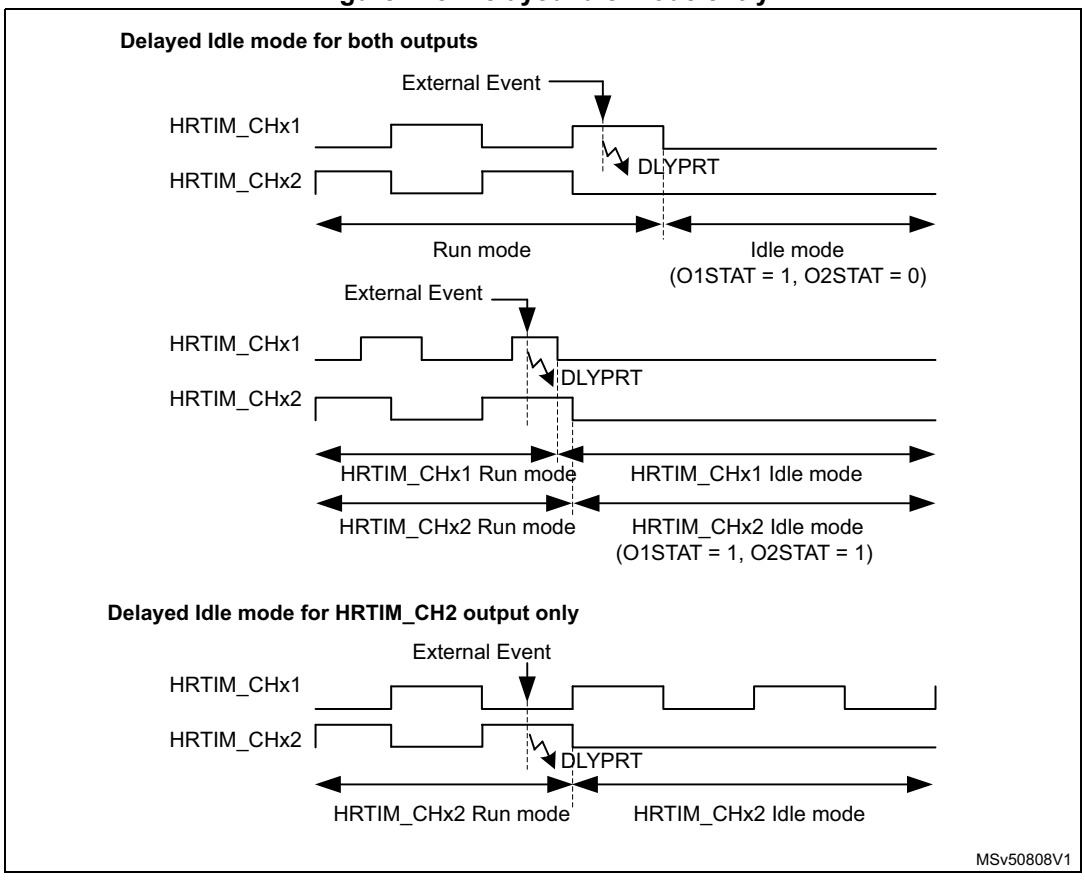
When the Delayed Idle mode is triggered, the output states can be determined using O1STAT and O2STAT in HRTIM_TIMxISR. Both status bits are updated even if the delayed idle is applied to a single output. When the push-pull mode is enabled, the IPPSTAT flag in HRTIM_TIMxISR indicates during which period the delayed protection request occurred.

This mode is available whatever the timer operating mode (regular, push-pull, deadtime). It is available with some external events only:

- EEVF and EEVG for timer A, B and C
- EEVH and EEVI for timer D, E and F

The delayed protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxEN bit is reset, until the TxyOEN bits are set.

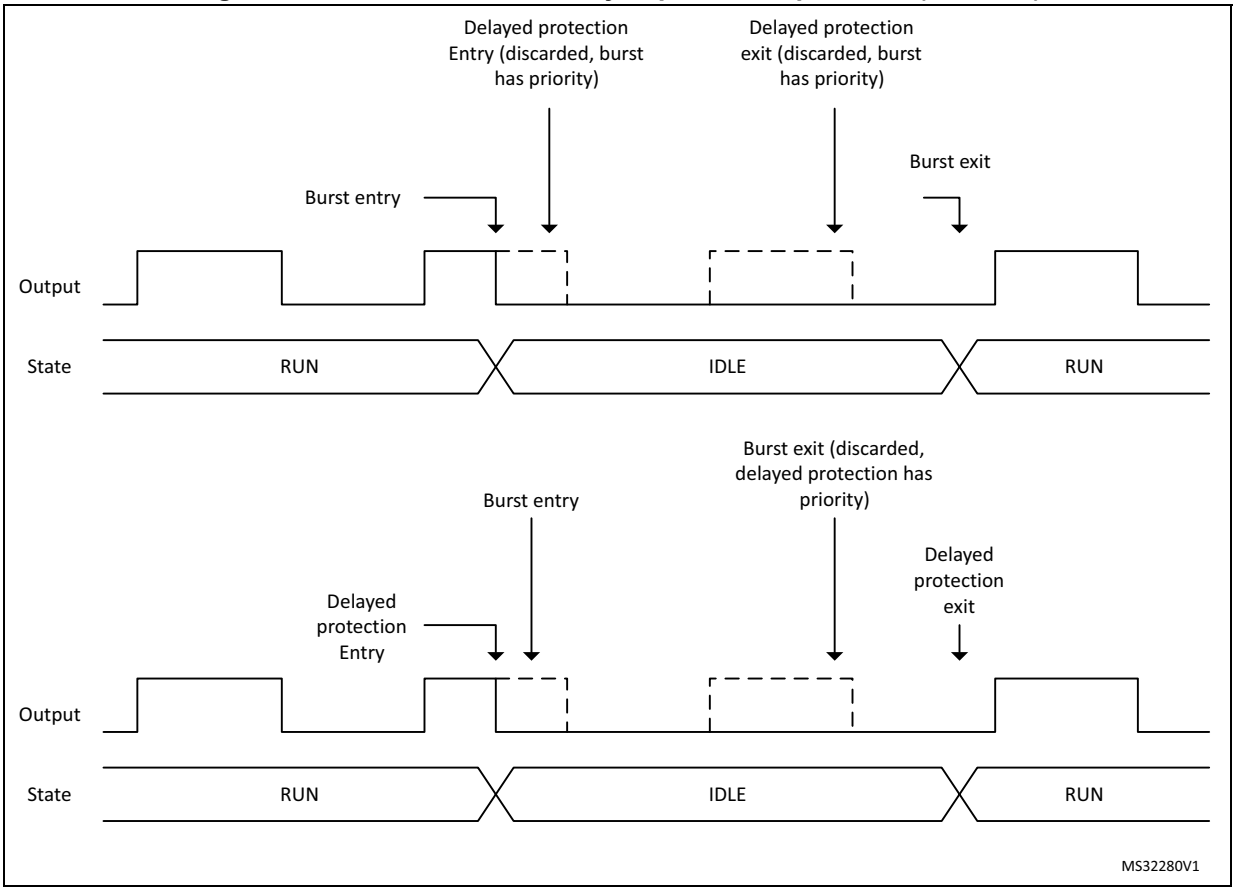
Figure 415. Delayed Idle mode entry



The delayed idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the delayed idle protection has been triggered. On the contrary, If the delayed protection is exited while the burst mode is active, the burst mode is resumed

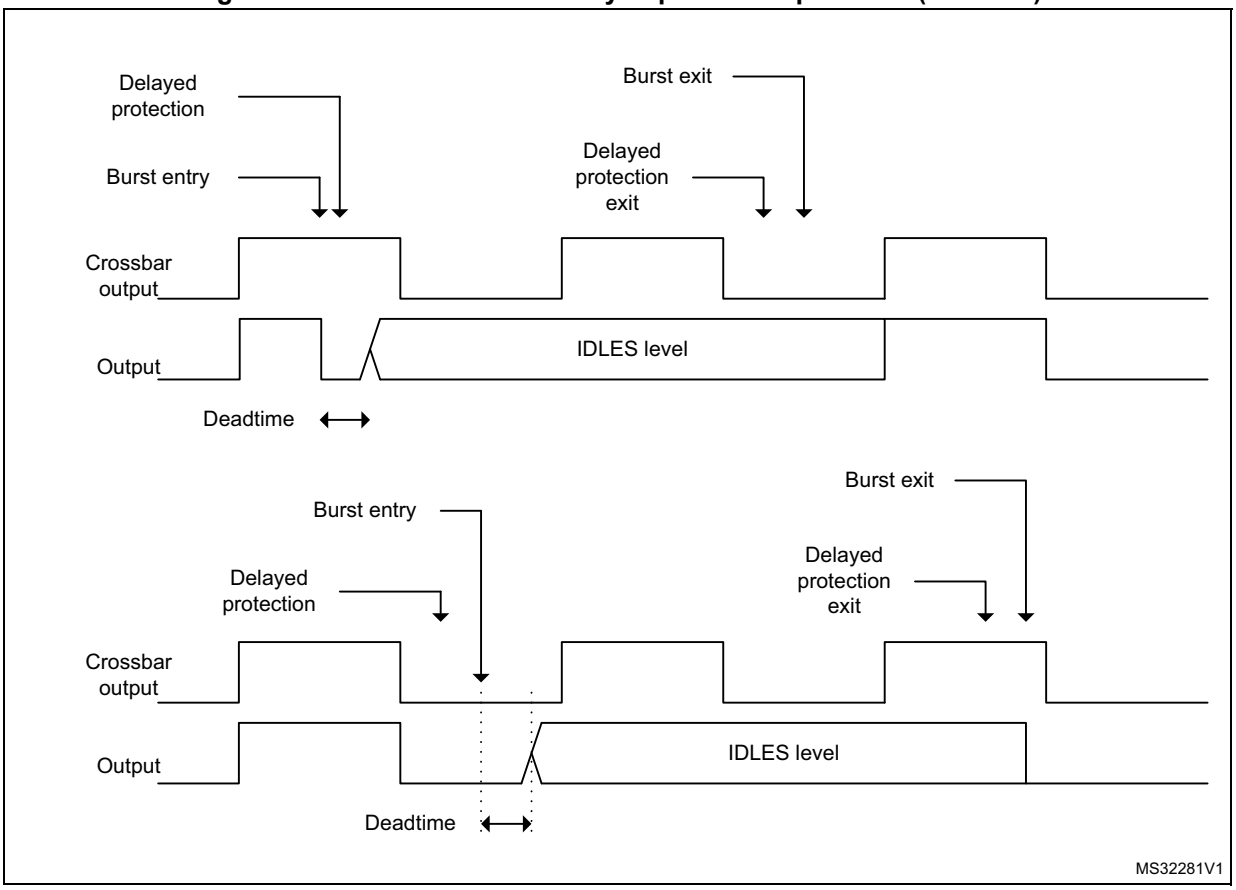
normally and the output is maintained in the idle state until the burst mode exits. *Figure 416* gives an overview of these different scenarios.

Figure 416. Burst mode and delayed protection priorities (DIDL = 0)



The same priorities are applied when the delayed burst mode entry is enabled (DIDL bit set), as shown on *Figure 417* below.

Figure 417. Burst mode and delayed protection priorities (DIDL = 1)



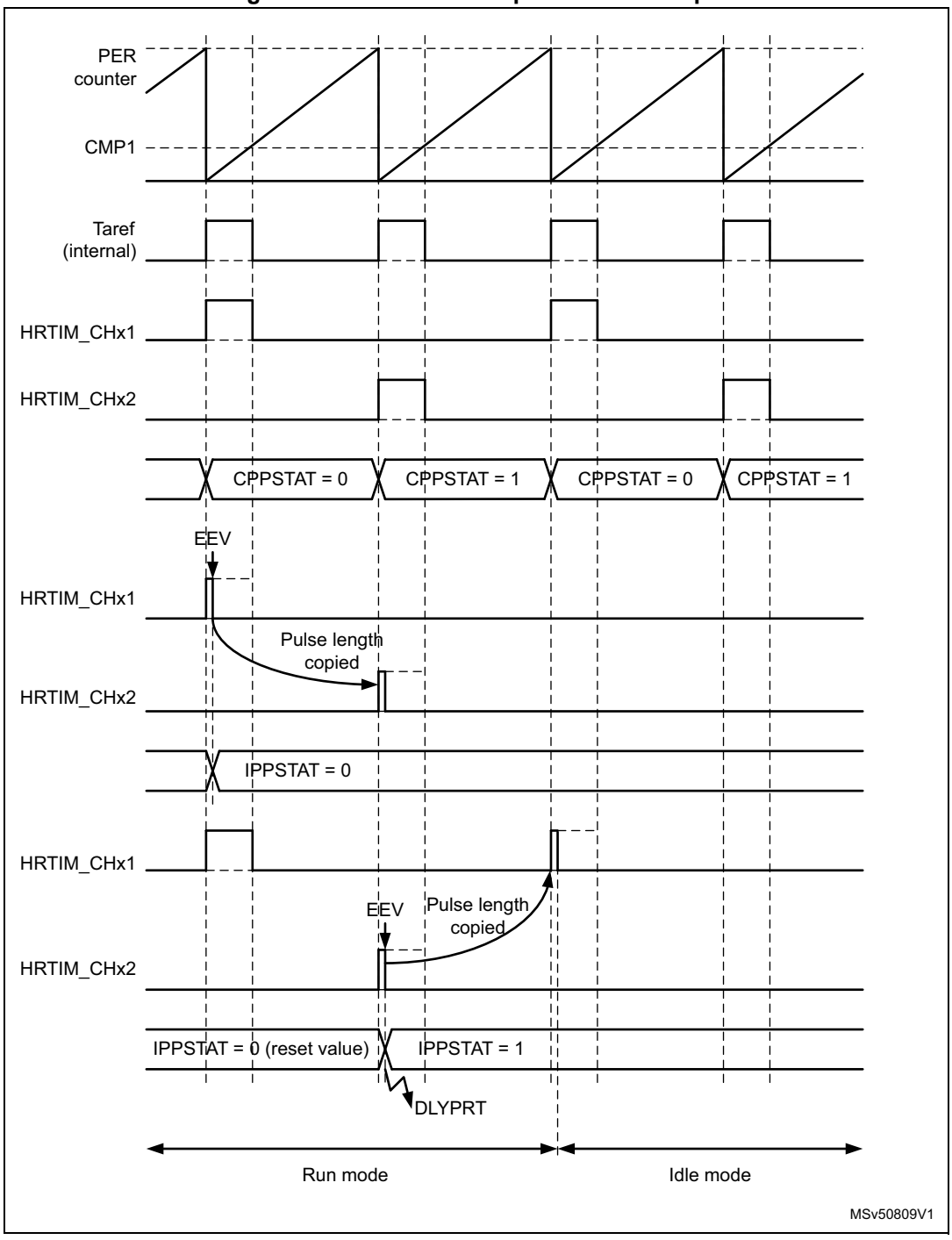
Balanced idle

Only available in push-pull mode, the balanced idle allows to have a balanced pulsewidth on the two outputs when one of the active pulse is shortened due to a protection. The pulsewidth, terminated earlier than programmed, is copied on the alternate output, then the two outputs are put in idle state, until the normal operation is resumed by software. This mode is enabled by writing x11 in DLYPRT[2:0] bitfield in HRTIM_OUTxR.

This mode is available with only 2 external events:

- EEVF and EEVG for timer A, B and C
- EEVH and EEVI for timer D, E and F

Figure 418. Balanced Idle protection example



When the balanced Idle mode is enabled, the selected external event triggers a capture of the counter value into the compare 4 active register (this value is not user-accessible). The push-pull is maintained for one additional period so that the shorten pulse can be repeated: a new output reset event is generated while the regular output set event is maintained.

The Idle mode is then entered and the output takes the level defined by IDLESx bits in the HRTIM_OUTxR register. The balanced idle mode entry is indicated by the DLYPRT flag,

while the IPPSTAT flag indicates during which period the external event occurred, to determine the sequence of shorten pulses (A1 then A2 or vice versa).

The timer operation is not interrupted (the counter continues to run).

To enable the balanced idle mode, it is necessary to have the following initialization:

- Timer operating in continuous mode (CONT = 1)
- Push-pull mode enabled
- HRTIM_CMP4xR must be set to 0 and the content transferred into the active register (for instance by forcing a software update)
- DELCMP4[1:0] bit field must be set to 00 (autodelayed mode disabled)
- DLYPRT[2:0] = x11 (delayed protection enable)

Note: The HRTIM_CMP4xR register must not be written during a balanced idle operation. The CMP4 event is reserved and cannot be used for another purpose.

In balanced idle mode, it is recommended to avoid multiple external events or software-based reset events causing an output reset. If such an event arrives before a balanced idle request within the same period, it causes the output pulses to be unbalanced (1st pulse length defined by the external event or software reset, while the 2nd pulse is defined by the balanced idle mode entry). The balanced idle mode is applicable only when a single pulse is generated per counting period, as on Figure 418: Balanced Idle protection example.

The minimum pulsewidth that can be handled in balanced idle mode is 4 f_{HRTIM} clock periods (0x80 if CKPSC[2:0] = 0, 0x40 if CKPSC[2:0] = 1, 0x20 if CKPSC[2:0] = 2,...).

If the capture occurs before the counter has reached this minimum value, the current pulse is extended up to 4 f_{HRTIM} clock periods before being copied into the secondary output. In any case, the pulsewidths are always balanced.

Tx1OEN and Tx2OEN bits are not affected by the balanced idle entry. To exit from balanced idle and resume the operation, it is necessary to overwrite Tx1OEN and Tx2OEN bits to 1 simultaneously. The output state changes on the first active transition following the output enable.

It is possible to resume operation similarly to the delayed idle entry. For instance, if the external event arrives while output 1 is active (delayed idle effective after output 2 pulse), the re-start sequence can be initiated for output 1 first. To do so, it is necessary to poll CPPSTAT bit in the HRTIM_TIMxISR register. Using the above example (IPPSTAT flag equal to 0), the operation is resumed when CPPSTAT bit is 0.

In order to have a specific re-start sequence, it is possible to poll the CPPSTAT to know which output is active first. This allows, for instance, to re-start with the same sequence as the idle entry sequence: if the external event arrives during output 1 active, the re-start sequence is initiated when the output 1 is active (CPPSTAT = 0).

Note: The balanced idle mode must not be disabled while a pulse balancing sequence is on-going. It is necessary to wait until the CMP4 flag is set, thus indicating that the sequence is completed, to reset the DLYPRTEN bit.

The balanced idle protection mode can be triggered only when the counter is enabled (TxCEN bit set). It remains active even if the TxCEN bit is reset, until TxyOEN bits are set.

Balanced idle can be used together with the burst mode under the following conditions:

- TxBM bit must be reset (counter clock maintained during the burst, see *Section 40.3.15*),
- No balanced idle protection must be triggered while the outputs are in a burst idle state.

The balanced idle mode has a higher priority than the burst mode: any burst mode exit request is discarded once the balanced idle protection has been triggered. On the contrary, if the delayed protection is exited while the burst mode is active, the burst mode is resumed normally.

Note: Although the output state is frozen in idle mode, a number of events are still generated on the auxiliary outputs (see Section 40.3.18) during the idle period following the delayed protection:
- Output set/reset interrupt or DMA requests
- External event filtering based on output signal
- Capture events triggered by set/reset

Balanced idle automatic resuming

The balanced Idle mode can be configured to have an automatic resuming of operation after a trigger.

Once the shorten pulse has been copied to the alternate output, the pulse width is reset to its original value and the timer resumes operation: the two outputs keep on being in RUN mode.

This is enabled by setting the BIAR bit in the HRTIM_OUTxR register.

This mode must be used only when the period in HRTIM_PERxR is greater than 6 periods of the f_{HRTIM} clock (0xC0 if CKPSC[2:0] = 0, 0x60 if CKPSC[2:0] = 1, 0x30 if CKPSC[2:0] = 2, ...).

Note: This bit is only significant if DLYPRT[2:0] = 011 or 111, it is ignored otherwise.

Note: In balanced idle automatic resuming mode, it is mandatory to set the IDLES state to inactive.

40.3.11 Register preload and update management

Most of HRTIM registers are buffered and can be preloaded if needed. Typically, this allows to prevent the waveforms from being altered by a register update not synchronized with the active events (set/reset).

When the preload mode is enabled, accessed registers are shadow registers. Their content is transferred into the active register after an update request, either software or synchronized with an event.

By default, PREEN bits in HRTIM_MCR and HRTIM_TIMxCR registers are reset and the registers are not preloaded: any write directly updates the active registers. If PREEN bit is reset while the timer is running and preload was enabled, the content of the preload registers is directly transferred into the active registers.

Each timing unit and the master timer have their own PREEN bit. If PRREN is set, the preload registers are enabled and transferred to the active register only upon an update event.

There are two options to initialize the timer when the preload feature is needed:

- Enable PREEN bit at the very end of the timer initialization to have the preload registers transferred into the active registers before the timer is enabled (by setting MCEN and TxCEN bits).
- Enable PREEN bit at any time during the initialization and force a software update immediately before starting.

Table 433 lists the registers which can be preloaded, together with a summary of available update events.

Table 433. HRTIM preloadable control registers and associated update sources

Timer	Preloadable registers	Preload enable	Update sources
Master timer	HRTIM_DIER HRTIM_MPER HRTIM_MREP HRTIM_MCMP1R HRTIM_MCMP2R HRTIM_MCMP3R HRTIM_MCMP4R	PREEN bit in HRTIM_MCR	Software Repetition event Burst DMA event Repetition event following a burst DMA event
Timer x x = A..F	HRTIM_TIMxDIER HRTIM_TIMxPER HRTIM_TIMxREP HRTIM_TIMxCMP1R HRTIM_TIMxCMP1CR HRTIM_TIMxCMP2R HRTIM_TIMxCMP3R HRTIM_TIMxCMP4R HRTIM_DTxR HRTIM_SETx1R HRTIM_RSTx1R HRTIM_SETx2R HRTIM_RSTx2R HRTIM_RSTxR	PREEN bit in HRTIM_TIMxCR	Software TIMx repetition event TIMx reset event Burst DMA event Update event from other timers (TIMy, master) Update event following a burst DMA event Update enable inputs hrtim_upd_en[3:1] Update event following an update enable input following an update event on hrtim_upd_en[3:1] inputs
HRTIM Common	HRTIM_ADC1R HRTIM_ADC2R HRTIM_ADC3R HRTIM_ADC4R	TIMx or master timer Update, depending on ADxUSRC[2:0] bits in HRTIM_CR1, if PREEN = 1 in the selected timer	

The master timer has 4 update options:

1. Software: writing 1 into MSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is cancelled.
2. Update done when the master counter rolls over and the master repetition counter is equal to 0. This is enabled when MREPU bit is set in HRTIM_MCR.
3. Update done once burst DMA is completed (see *Section 40.3.23* for details). This is enabled when BRSTDMA[1:0] = 01 in HRTIM_MCR. It is possible to have both MREPU=1 and BRSTDMA=01.
   Note: The update can take place immediately after the end of the burst sequence if SWU bit is set (i.e. forced update mode). If SWU bit is reset, the update is done on the next update event following the end of the burst sequence.
4. Update done when the master counter rolls over following a burst DMA completion. This is enabled when BRSTDMA[1:0] = 10 in HRTIM_MCR.

An interrupt or a DMA request can be generated by the master update event.

Each timer (TIMA..F) can also have the update done as follows:

- By software: writing 1 into TxSWU bit in HRTIM_CR2 forces an immediate update of the registers. In this case, any pending hardware update request is canceled.
- Update done when the counter rolls over and the repetition counter is equal to 0. This is enabled when TxREPU bit is set in HRTIM_TIMxCR.
- Update done when the counter is reset or rolls over in continuous mode. This is enabled when TxRSTU bit is set in HRTIM_TIMxCR. This is used for a timer operating in single-shot mode, for instance.
- Update done once a burst DMA is completed. This is enabled when UPDGAT[3:0] = 0001 in HRTIM_TIMxCR.
- Update done on the update event following a burst DMA completion (the event can be enabled with TxRSTU, TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0010 in HRTIM_TIMxCR.
- Update done when receiving a request on hrtim_upd_en[3:1]. This is enabled when UPDGAT[3:0] = 0011, 0100, 0101 in HRTIM_TIMxCR.
- Update done on the update event following a request on hrtim_upd_en[3:1] (the event can be enabled with TxRSTU, TxREPU, MSTU or TxU). This is enabled when UPDGAT[3:0] = 0110, 0111, 1000 in HRTIM_TIMxCR
- Update done synchronously with any other timer or master update (for instance TIMA can be updated simultaneously with TIMB). This is used for converters requiring several timers, and is enabled by setting bits MSTU and TxU in HRTIM_TIMxCR register.

The update enable inputs hrtim_upd_en[3:1] allow to have an update event synchronized with on-chip events coming from the general-purpose timers. These inputs are rising-edge sensitive.

Table 412 lists the connections between update enable inputs and the on-chip sources.

This allows to synchronize low frequency update requests with high-frequency signals (for instance an update on the counter roll-over of a 100 kHz PWM that has to be done at a 100 Hz rate).

Note: The update events are synchronized to the prescaler clock when CKPSC[2:0] > 5.

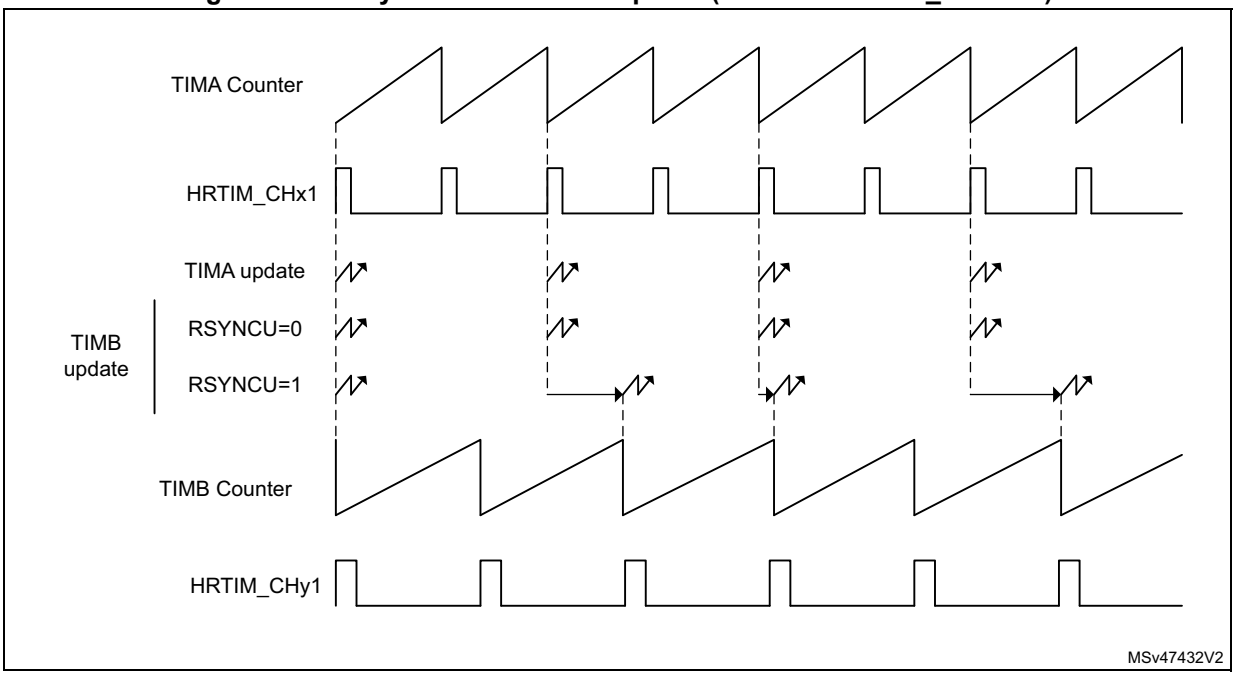
The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) can either be taken into account immediately or re-synchronized with the timers reset/roll-over event. This is done with the RSYNCU bit in the HRTIM_TIMxCR register, as show on *Figure 419* below):

- RSYNCU = 0: The update coming from adjacent timers is taken into account immediately
- RSYNCU = 1: The update coming from adjacent timers is taken into account on the following reset/roll-over event.

The RSYNCU bit is significant only when UPDGAT[3:0] = 0000, it is ignored otherwise.

An interrupt or a DMA request can be generated by the Timx update event.

Figure 419. Resynchronized timer update (TAU=1 in HRTIM_TIMBCR)



MUDIS and TxUDIS bits in the HRTIM_CR1 register allow to temporarily disable the transfer from preload to active registers, whatever the selected update event. This allows to modify several registers in multiple timers. The register update event takes place once these bits are reset, on the next coming update event.

MUDIS and TxUDIS bits are all grouped in the same register. This allows the update of multiple timers (not necessarily synchronized) to be disabled and resumed simultaneously.

The following example is a practical use case. A first power converter is controlled with the master, TIMB and TIMC. TIMB and TIMC must be updated simultaneously with the master timer repetition event. A second converter works in parallel with TIMA, TIMD and TIME, and TIMD, TIME must be updated with TIMA repetition event.

First converter

In HRTIM_MCR, MREPU bit is set: the update occurs at the end of the master timer counter repetition period. In HRTIM_TIMBCR and HRTIM_TIMCCR, MSTU bits are set to have TIMB and TIMC timers updated simultaneously with the master timer.

When the power converter set-point has to be adjusted by software, MUDIS, TBUDIS and TCUDIS bits of the HRTIM_CR register must be set prior to write accessing registers to update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, MUDIS, TBUDIS and TCUDIS bits must be reset. The transfer from preload to active registers is done as soon as the master repetition event occurs.

Second converter

In HRTIM_TIMACR, TAREPU bit is set: the update occurs at the end of the timer A counter repetition period. In HRTIM_TIMDCR and HRTIM_TIMECR, TAU bits are set to have TIMD and TIME timers updated simultaneously with timer A.

When the power converter set-point has to be adjusted by software, TAUDIS, TDUDIS and TEUDIS bits of the HRTIM_CR register must be set prior to write accessing the registers to

update the values (for instance the compare values). From this time on, any hardware update request is ignored and the preload registers can be accessed without any risk to have them transferred into the active registers. Once the software processing is over, TAUDIS, TDUDIS and TEUDIS bits can be reset: the transfer from preload to active registers is done as soon as the timer A repetition event occurs.

40.3.12 PWM mode with “greater than” comparison

A specific no-latency update mode is available for PWM signals generated with the CMP1 and CMP3 registers. It allows to have a new duty cycle value applied as soon as possible within the PWM cycle, without having to wait the completion of the current PWM period. This reduces the overall delay time in software control loops. As shown on *Figure 420* below, this eventually allows to have:

- An early turn-off of the output if the new compare value is below the current counter value and the current compare value is above the counter, at the time the new value is written.
- An early turn-on of the output, re-enabling the output if the new compare value is above the counter value and the current compare value is below the counter, at the time the new value is written.

The output signal is left unchanged when the new compare value and current compare value are both below the counter.

This feature is only available for CMP1 or CMP3 RESET events, and is enabled using the GTCMP1 and GTCMP3 enable bits in the HRTIM_TIMxCR2 register.

The preload mechanism is inactive for a compare register when the corresponding GTCMPx bit is set, whatever the PREEN bit value. This mode is intended to have the new compare value taken into account as soon as possible after a new value write, without waiting for the preload to active register transfer.

These bits are defining the compare 1 and compare 3 operating modes as following

- GTCMPx = 0: the compare x event is generated when the counter is equal to the compare value (compare match mode). If the compare value is changed on-the-fly, the compare event may not be generated.
- GTCMPx = 1: the compare x event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

The “greater than” compare mode causes the crossbar to act differently depending on comparison result. Let’s consider the CMP1 event is doing an output RESET. When the new compare value is written, two cases are considered

- If the new compare value is below the counter value, the RESET event is issued and can eventually cause an early turn-OFF
- If the new compare value is above the counter value, a SET event is generated so as to re-arm the output value before it is actually RESET when the counter exceeds the counter value (early turn-ON).

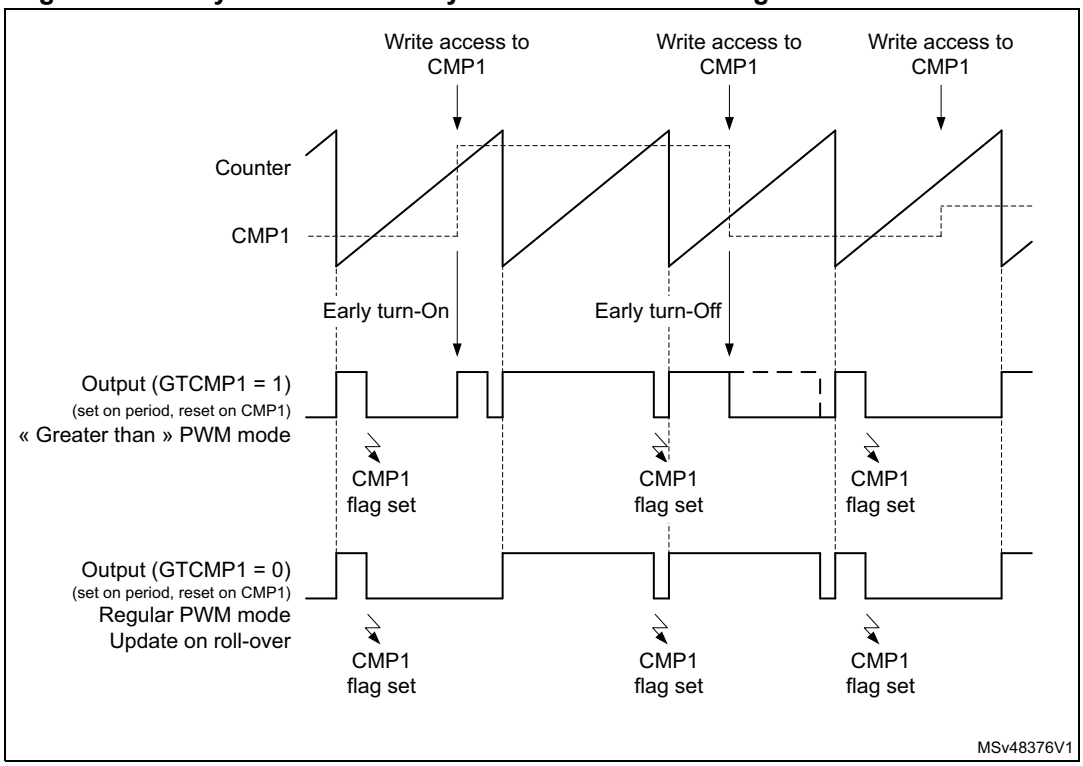
The “greater than” compare mode is supported for both SET and RESET actions.

The “greater than” compare mode must only be used for the following configuration:

1. In the fixed frequency configuration, the period event must trigger the output set and the “greater than” compare triggers the output reset (or vice versa the period must trigger the reset if the “greater-than” compare triggers the set).
2. For variable frequency configuration, the event selected as counter reset source must also be selected as set or reset source for the timer output (opposite direction as the “greater than” compare event).

Note: The “greater-than” modes must not be used when the CMP1 and/or CMP3 modes are controlled by hardware in half and interleaved modes.

Figure 420. Early turn-ON and early turn-OFF behavior in “greater than” PWM mode



The immediate update mode implies that the content of the preload register is transferred into the active register at the very same time the register is written. When GTCMP1 and/or GTCMP3 bits are set, their respective preload mechanism is disabled (for HRTIM_TIMxCMP1 and/or HRTIM_TIMxCMP3 registers), whatever the PREEN bit value.

Note: The compare interrupt flags (CMP1 and CMP3 in HRTIM_TIMxISR) are not generated in case of late turn-ON and early turn-OFF, as shown on Figure 420.

Note: The “Greater than” comparison must not be done on both CMP1 and CMP3 for the same output (GTCMP1 and GTCMP3 bits must not be set simultaneously).

40.3.13 Events propagation within or across multiple timers

The HRTIM offers many possibilities for cascading events or sharing them across multiple timing units, including the master timer, to get full benefits from its modular architecture. These are key features for converters requiring multiple synchronized outputs.

This section summarizes the various options and specifies whether and how an event is propagated within the HRTIM.

TIMx update triggered by the master timer update

The sources listed in *Table 434* are generating a master timer update. The table indicates if the source event can be used to trigger a simultaneous update in any of TIMx timing units.

Operating condition: MSTU bit is set in HRTIM_TIMxCR register.

Table 434. Master timer update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	BRSTDMA[1:0] = 01	No	Must be done in TIMxCR (UPDGAT[3:0] = 0001)
Roll-over event following a burst DMA end	BRSTDMA[1:0] = 10	Yes	-
Repetition event caused by a counter roll-over	MREPU = 1	Yes	-
Repetition event caused by a counter reset (from HRTIM_SCIN or software)		No	-
Software update	MSWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx update triggered by the TIMy update

The sources listed in *Table 435* are generating a TIMy update. The table indicates if the given event can be used to trigger a simultaneous update in another or multiple TIMx timers.

Operating condition: TyU bit set in HRTIM_TIMxCR register (source = TIMy and destination = TIMx).

Table 435. TIMx update event propagation

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0001)
Update caused by the update enable inputs hrtim_upd_en[3:1]	UPDGAT[3:0] = 0011, 0100, 0101	No	Must be done directly in HRTIM_TIMxCR (UPDGAT[3:0] = 0011, 0100, 0101
Master update	MSTU = 1 in HRTIM_TIMyCR	No	Must be done with MSTU = 1 in HRTIM_TIMxCR
Another TIMx update (TIMz>TIMy>TIMx)	TzU=1 in HRTIM_TIMyCR TyU=1 in TIMxCR	No	Must be done with TzU=1 in HRTIM_TIMxCR TzU=1 in HRTIM_TIMyCR
Repetition event caused by a counter roll-over	TyREPU = 1	Yes	-
Repetition event caused by a counter reset	TyREPU = 1	-	Refer to counter reset cases below
Counter roll-over	TyRSTU = 1	Yes	-

Table 435. TIMx update event propagation (continued)

Source	Condition	Propagation	Comment
Counter software reset	TyRST=1 in HRTIM_CR2	No	Can be done simultaneously with update in HRTIM_CR2 register
Counter reset caused by a TIMz compare	TIMzCMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by external events	EXTEVNTn in HRTIM_RSTyR	Yes	-
Counter reset caused by a master compare or a master period	MSTCMPn or MSTPER in HRTIM_RSTyR	Yes	-
Counter reset caused by a TIMy compare	CMPn in HRTIM_RSTyR	Yes	-
Counter reset caused by an update	UPDT in HRTIM_RSTyR	No	Propagation would result in a lock-up situation (update causing reset causing update)
Counter reset caused by HRTIM_SCIN	SYNCRSTy in HRTIM_TIMyCR	No	-
Software update	TySWU = 1	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update

TIMx counter reset causing a TIMx update

Table 436 lists the counter reset sources and indicates whether they can be used to generate an update.

Operating condition: TxRSTU bit in HRTIM_TIMxCR register.

Table 436. Reset events able to generate an update

Source	Condition	Propagation	Comment
Counter roll-over	-	Yes	-
Update event	UPDT in HRTIM_RSTxR	No	Propagation would result in a lock-up situation (update causing a reset causing an update)
External event	EXTEVNTn in HRTIM_RSTxR	Yes	-
TIMy compare	TIMyCMPn in HRTIM_RSTxR	Yes	-
Master compare	MSTCMPn in HRTIM_RSTxR	Yes	-
Master period	MSTPER in HRTIM_RSTxR	Yes	-
Compare 2 and 4	CMPn in HRTIM_RSTxR	Yes	-
Software	TxRST=1 in HRTIM_CR2	Yes	-
HRTIM_SCIN	SYNCRSTx in HRTIM_TIMxCR	Yes	-

TIMx update causing a TIMx counter reset

Table 437 lists the update event sources and indicates whether they can be used to generate a counter reset.

Operating condition: UPDT bit set in HRTIM_RSTxR.

Table 437. Update event propagation for a timer reset

Source	Condition	Propagation	Comment
Burst DMA end	UPDGAT[3:0] = 0001	Yes	-
Update caused by the update enable inputs hrtim_upd_en[3:1]	UPDGAT[3:0] = 0011, 0100, 0101	Yes	-
Master update caused by a roll-over after a burst DMA	MSTU = 1 in HRTIM_TIMxCR BRSTDMA[1:0] = 10 in HRTIM_MCR	Yes	-

Table 437. Update event propagation for a timer reset (continued)

Source	Condition	Propagation	Comment
Master update caused by a repetition event following a roll-over	MSTU = 1 in HRTIM_TIMxCR MREPU = 1 in HRTIM_MCR	Yes	-
Master update caused by a repetition event following a counter reset (software or due to HRTIM_SCIN)		No	-
Software triggered master timer update	MSTU = 1 in HRTIM_TIMxCR MSWU = 1 in HRTIM_CR2	No	All software update bits (TxSWU) are grouped in the HRTIM_CR2 register and can be used for a simultaneous update
TIMy update caused by a TIMy counter roll-over	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by a TIMy repetition event	TyU = 1 in HRTIM_TIMxCR TyREPU = 1 in HRTIM_TIMyCR	Yes	-
TIMy update caused by an external event or a TIMy compare (through a TIMy reset)	TyU = 1 in HRTIM_TIMxCR TyRSTU = 1 in HRTIM_TIMyCR EXTEVNTn or CMP4/2 in HRTIM_RSTyCR	Yes	-
TIMy update caused by sources other than those listed above	TyU = 1 in HRTIM_TIMxCR	No	-
Repetition event following a roll-over	TxREPU = 1 in HRTIM_TIMxCR	Yes	-
Repetition event following a counter reset		No	-
Timer reset	TxRSTU = 1 in HRTIM_TIMxCR	No	Propagation would result in a lock-up situation (reset causing an update causing a reset)
Software	TxSWU in HRTIM_CR2	No	-

40.3.14 Output management

Each timing unit controls a pair of outputs. The outputs have three operating states:

- RUN: this is the main operating mode, where the output can take the active or inactive level as programmed in the crossbar unit.
- IDLE: this state is the default operating state after an HRTIM reset, when the outputs are disabled by software or during a burst mode operation (where outputs are temporarily disabled during a normal operating mode; refer to *Section 40.3.15* for more details). It is either permanently active or inactive.
- FAULT: this is the safety state, entered in case of a shut-down request on FAULTx inputs. It can be permanently active, inactive, or Hi-Z.

The output status is indicated by TxyOEN bit in HRTIM_OENR register and TxyODS bit in HRTIM_ODSR register, as in *Table 438*.

Table 438. Output state programming, x= A..F, y = 1 or 2

TxyOEN (control/status) (set by software, cleared by hardware)	TxyODS (status)	Output operating state
1	x	RUN
0	0	IDLE
0	1	FAULT

TxyOEN bit is both a control and a status bit: it must be set by software to have the output in RUN mode. It is cleared by hardware when the output goes back in IDLE or FAULT mode. When TxyOEN bit is cleared, TxyODS bit indicates whether the output is in the IDLE or FAULT state. A third bit in the HRTIM_ODISR register is used to disable the output by software.

Figure 421. Output management overview

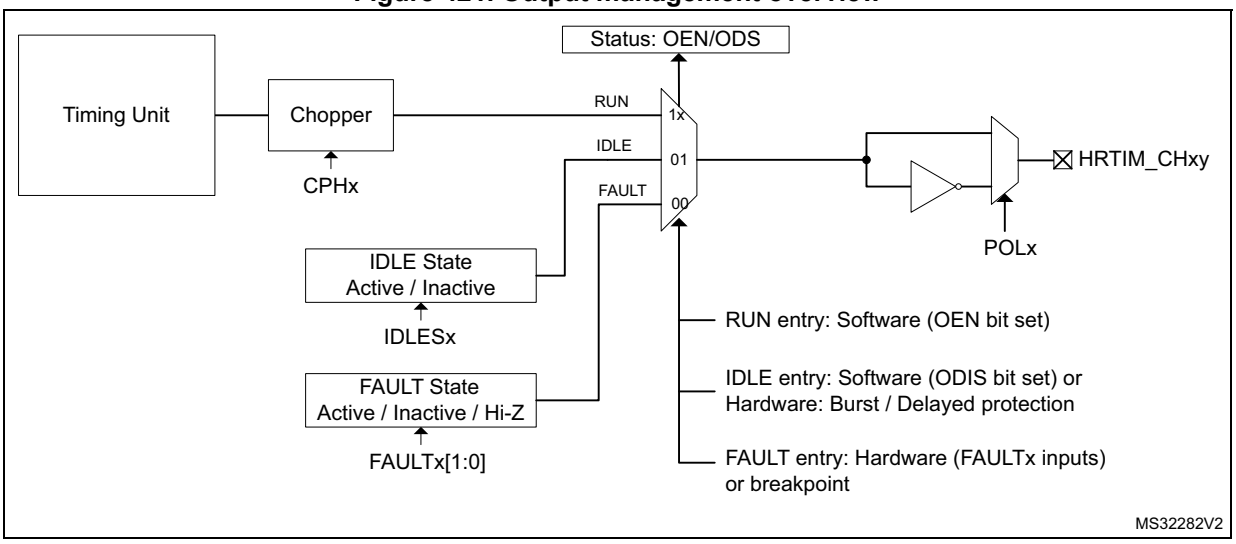
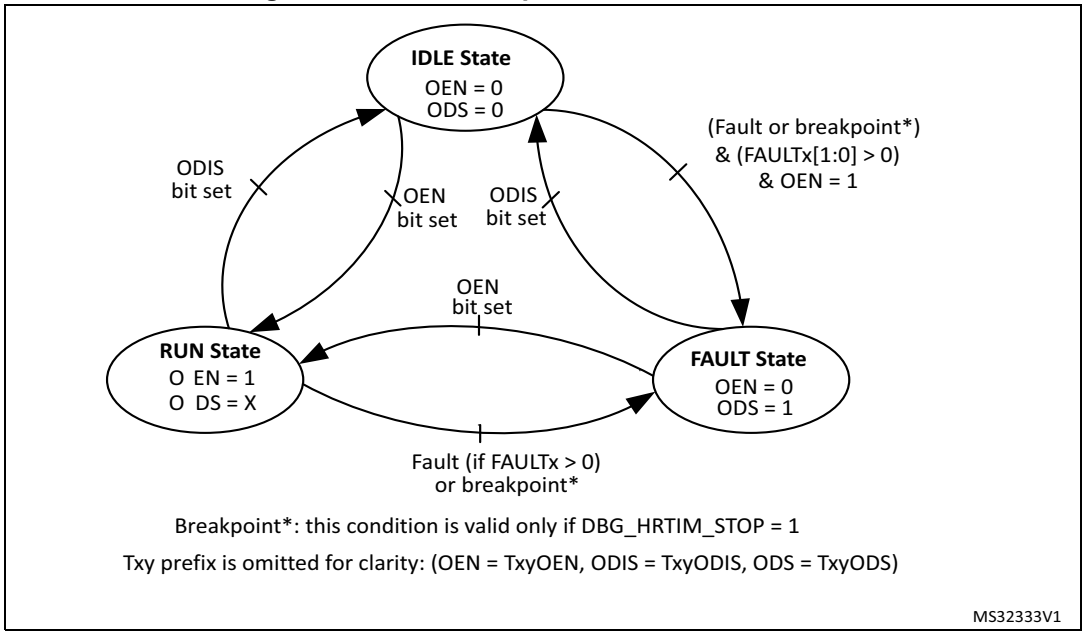


Figure 422 summarizes the bit values for the three states and how the transitions are triggered. Faults can be triggered by any external or internal fault source, as listed in

Section 40.3.17, while the IDLE state can be entered when the burst mode or delayed protections are active.

Figure 422. HRTIM output states and transitions



The FAULT and IDLE levels are defined as active or inactive. Active (or inactive) refers to the level on the timer output that causes a power switch to be closed (or opened for an inactive state).

The IDLE state has the highest priority: the transition FAULT → IDLE is possible even if the FAULT condition is still valid, triggered by ODIS bit set.

The FAULT state has priority over the RUN state: if TxyOEN bit is set simultaneously with a fault event, the FAULT state is entered. The condition is given on the transition IDLE → FAULT, as in *Figure 422*: fault protection needs to be enabled (FAULTx[1:0] bits = 01, 10, 11) and the Txy OEN bit set with a fault active (or during a breakpoint if DBG_HRTIM_STOP = 1).

The output polarity is programmed using POLx bits in HRTIM_OUTxR. When POLx = 0, the polarity is positive (output active high), while it is active low in case of a negative polarity (POLx = 1). Practically, the polarity is defined depending on the power switch to be driven (PMOS versus NMOS) or on a gate driver polarity.

The output level in the FAULT state is configured using FAULTx[1:0] bits in HRTIM_OUTxR, for each output, as follows:

- 00: output never enters the fault state and stays in RUN or IDLE state
- 01: output at active level when in FAULT
- 10: output at inactive level when in FAULT
- 11: output is tri-stated when in FAULT. The safe state must be forced externally with pull-up or pull-down resistors, for instance.

Note: FAULTx[1:0] bits must not be changed as long as the outputs are in FAULT state.

The level of the output in IDLE state is configured using IDLESx bit in HRTIM_OUTxR, as follows:

- 0: output at inactive level when in IDLE
- 1: output at active level when in IDLE

When TxyOEN bit is set to enter the RUN state, the output is immediately connected to the crossbar output. If the timer clock is stopped, the level is either inactive (after an HRTIM reset) or corresponds to the RUN level (when the timer is stopped and the output disabled).

During the HRTIM initialization, the output level can be prepositioned prior to have it in RUN mode, using the software forced output set and reset in the HRTIM_SETx1R and HRTIM_RSTx1R registers.

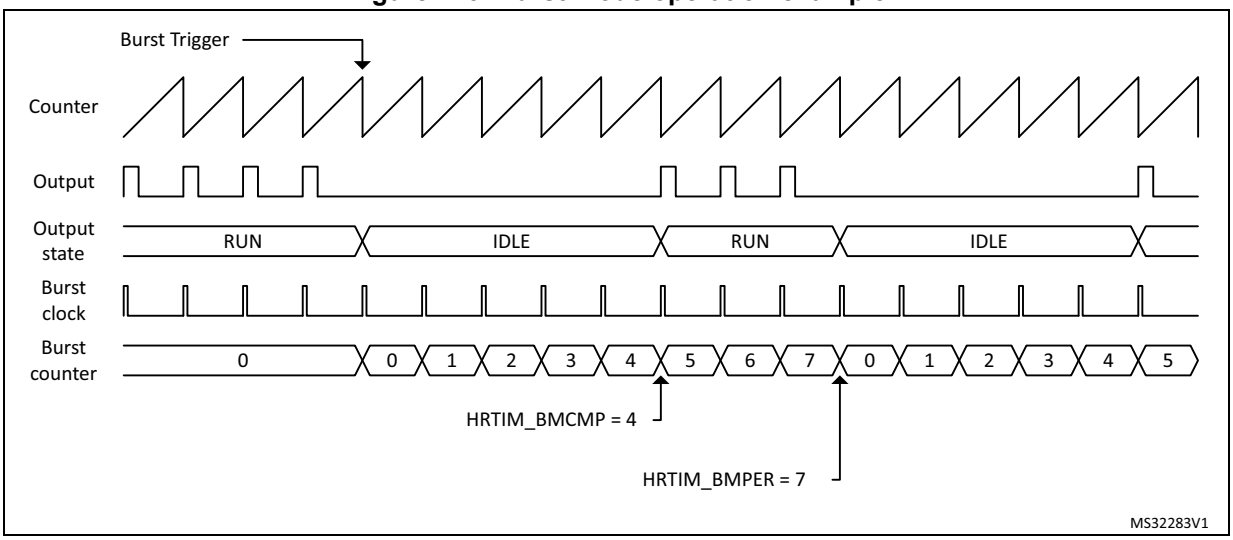
40.3.15 Burst mode controller

The burst mode controller is used to have the outputs alternatively in IDLE and RUN state, by hardware, so as to skip some switching periods with a programmable periodicity and duty cycle.

Burst mode operation is of common use in power converters when operating under light loads. It can significantly increase the efficiency of the converter by reducing the number of transitions on the outputs and the associated switching losses.

When operating in burst mode, one or a few pulses are outputted followed by an idle period equal to several counting periods, typically, where no output pulses are produced, as shown in the example on *Figure 423*.

Figure 423. Burst mode operation example



The burst mode controller consists of:

- A counter that can be clocked by various sources, either within or outside the HRTIM (typically the end of a PWM period).
- A compare register to define the number of idle periods: HRTIM_BMCMP.
- A period register to define the burst repetition rate (corresponding to the sum of the idle and run periods): HRTIM_BMPER.

The burst mode controller is able to take over the control of any of the 10 PWM outputs. The state of each output during a burst mode operation is programmed using IDLESx and IDLEMx bits in the HRTIM_OUTxR register, as in *Table 439*.

Table 439. Timer output programming for burst mode

IDLEMx	IDLESx	Output state during burst mode
0	X	No action: the output is not affected by the burst mode operation.
1	0	Output inactive during the burst
1	1	Output active during the burst

Note: IDLEMx bit must not be changed while the burst mode is active.

The burst mode controller only acts on the output stage. A number of events are still generated during the idle period:

- Output set/reset interrupt or DMA requests
- External event filtering based on Tx2 output signal
- Capture events triggered by output set/reset

During the burst mode, neither start nor reset events are generated on the hrtim_out_sync[2:1] output, even if TxBM bit is set.

Operating mode

It is necessary to have the counter enabled (TxCEN bit set) before using the burst mode on a given timing unit. The burst mode is enabled with BME bit in the HRTIM_BMCR register.

It can operate in continuous or single-shot mode, using BMOM bit in the HRTIM_BMCR register. The continuous mode is enabled when BMOM = 1. The burst operation is maintained until BMSTAT bit in HRTIM_BMCR is reset to terminate it.

In single-shot mode (BMOM = 0), the idle sequence is executed once, following the burst mode trigger, and the normal timer operation is resumed immediately after.

The duration of the idle and run periods is defined with a burst mode counter and two registers. The HRTIM_BMCMPR register defines the number of counts during which the selected timer(s) are in an idle state (idle period). HRTIM_BMPER defines the overall burst mode period (sum of the idle and run periods). Once the initial burst mode trigger has occurred, the idle period length is HRTIM_BMCMPR+1, the overall burst period is HRTIM_BMPER+1.

Note: The burst mode period must not be less than or equal to the deadtime duration defined with DTRx[8:0] and DTFx[8:0] bitfields.

The counters of the timing units and the master timer can be stopped and reset during the burst mode operation. HRTIM_BMCR holds six control bits for this purpose: MTBM (master) and TABM..TEBM for timer A..E.

When MTBM or TxBM bit is reset, the counter clock is maintained. This is used to keep a phase relationship with other timers in multiphase systems, for instance.

When MTBM or TxBM bit is set, the corresponding counter is stopped and maintained in reset state during the burst idle period. This is used to have the timer restarting a full period when exiting from idle. If SYNCSRC[1:0] = 00 or 10 (synchronization output on the master

start or timer A start), a pulse is sent on the HRTIM_SCOUT output when exiting the burst mode.

Note: TxBM bit must not be set when the balanced idle mode is active (DLYPRT[1:0] = 0x11).

Burst mode clock

The burst mode controller counter can be clocked by several sources, selected with BMCLK[3:0] bits in the HRTIM_BMCR register:

- BMCLK[3:0] = 0000 to 0101: master timer and TIMA..E reset/roll-over events. This is used to have burst mode idle and run periods aligned with the timing unit counting period (both in free-running and counter reset mode).
- BMCLK[3:0] = 0110 to 1001: The clocking is provided by the hrtim_bm_ck[4:1] inputs connected to general purpose timers, as in *Table 413*. In this case, the burst mode idle and run periods are not necessarily aligned with timing unit counting period (a pulse on the output may be interrupted, resulting in a waveform with modified duty cycle for instance.
- BMCLK[3:0] = 1010: The f_{HRTIM} clock prescaled by a factor defined with BMPRSC[3:0] bits in HRTIM_BMCR register. In this case, the burst mode idle and run periods are not necessarily aligned with the timing unit counting period (a pulse on the output may be interrupted, resulting in a waveform with a modified duty cycle, for instance.

The pulse width on TIMx OC output must be at least N f_{HRTIM} clock cycles long to be detected by the HRTIM burst mode controller.

Burst mode triggers

To trigger the burst operation, 32 sources are available and are selected using the HRTIM_BMTRGR register:

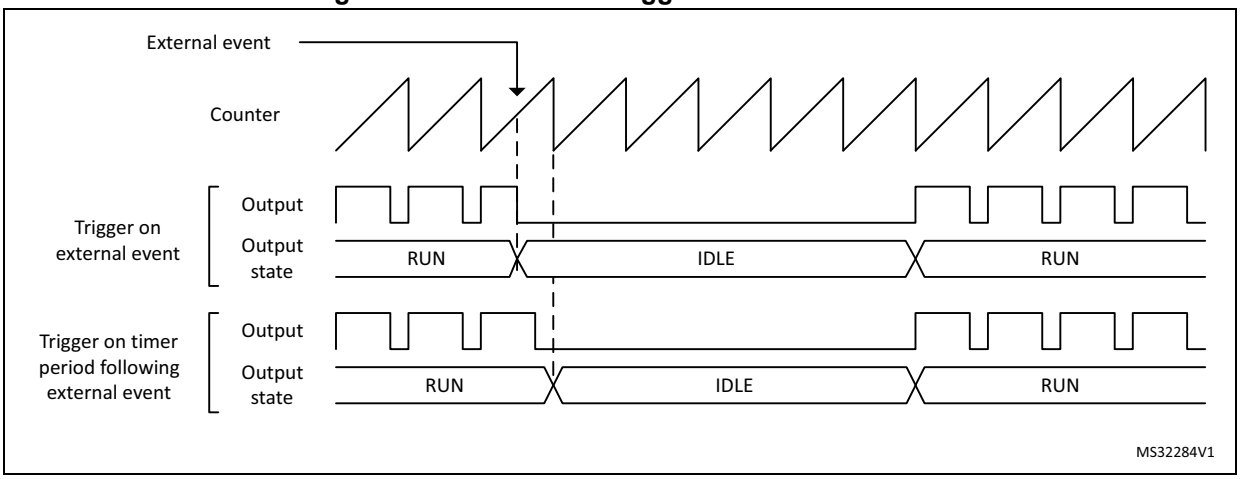
- Software trigger (set by software and reset by hardware)
- 6 master timer events: repetition, reset/roll-over, compare 1 to 4
- 5 x 4 events from timers A..F: repetition, reset/roll-over, compare 1 and 2
- External event G (including TIMA event filtering) and H (including TIMD event filtering)
- Timer A period following external event G (including TIMA event filtering)
- Timer D period following external event H (including TIMD event filtering)
- An on-chip event on the hrtim_bm_trg input (connected to the general-purpose timer TRGO output), see *Table 410* for details.

These sources can be combined to have multiple concurrent triggers.

Burst mode is not retriggerable. In continuous mode, new triggers are ignored until the burst mode is terminated, while in single-shot mode, the triggers are ignored until the current burst completion including run periods (HRTIM_BMPER+1 cycles). This is also valid for software trigger (the software bit is reset by hardware even if it is discarded).

Figure 424 shows how the burst mode is started in response to an external event, either immediately or on the timer period following the event.

Figure 424. Burst mode trigger on external event



For TAEEVG and TDEEVH combined triggers (trigger on a timer period following an external event), the external event detection is always active, regardless of the burst mode programming and the ongoing burst operation:

- When the burst mode is enabled (BME=1) or the trigger is enabled (TAEEVG or TDEEVH bit set in the HRTIM_BMTRGR register) in between the external event and the timer period event, the burst is triggered.
- The single-shot burst mode is retriggered even if the external event occurs before the burst end (as long as the corresponding period happens after the burst).

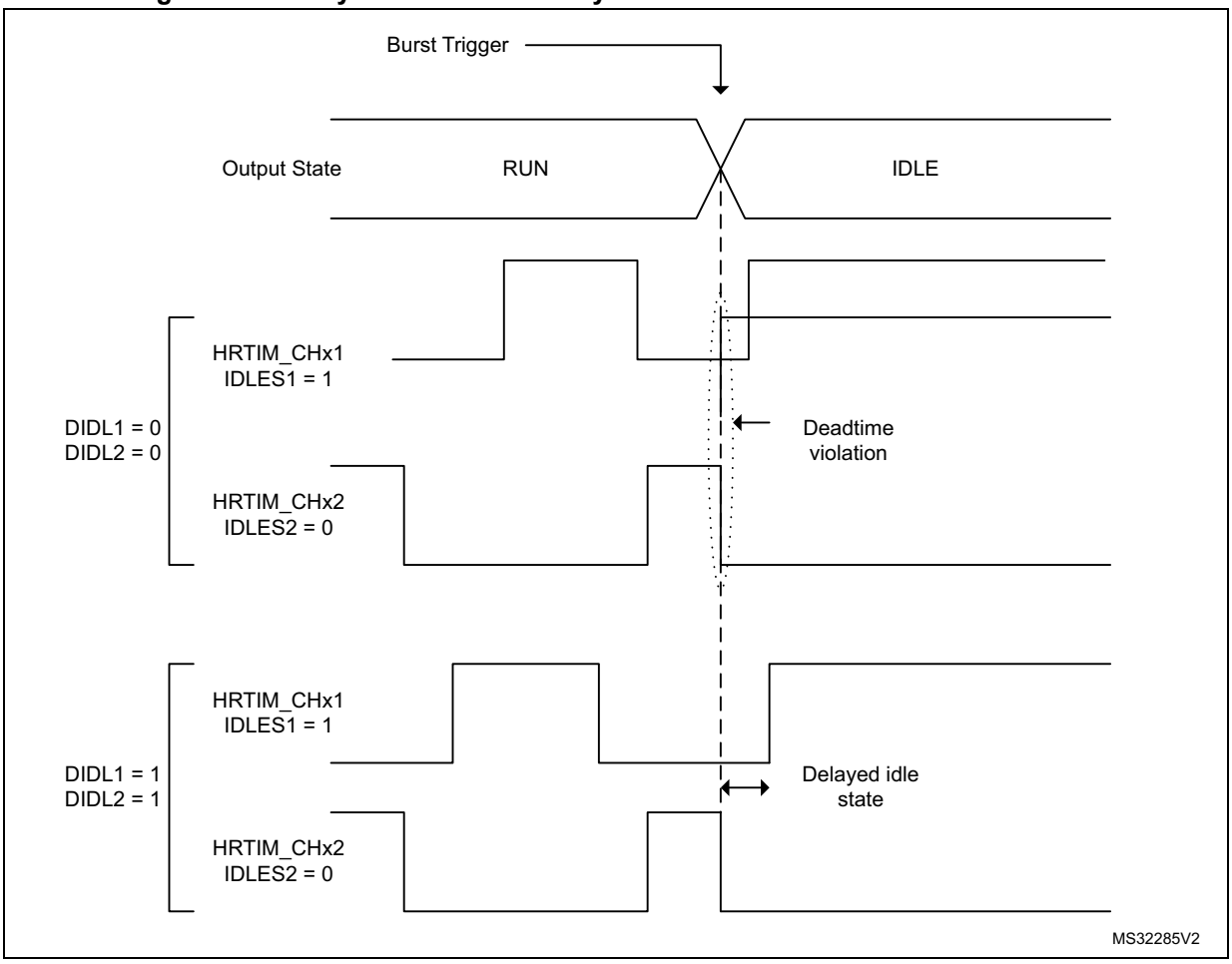
Note: TAEEVG and TDEEVH triggers are valid only after a period event. If the counter is reset before the period event, the pending hrtim_eev7 or hrtim_eev8 event is discarded.

Burst mode delayed entry

By default, the outputs are taking their idle level (as per IDLES1 and IDLES2 setting) immediately after the burst mode trigger.

It is also possible to delay the burst mode entry and force the output to an inactive state during a programmable period before the output takes its idle state. This is useful when driving two complementary outputs, one of them having an active idle state, to avoid a deadtime violation as shown on *Figure 425*. This prevents any risk of shoot through current in half-bridges, but causes a delayed response to the burst mode entry.

Figure 425. Delayed burst mode entry with deadtime enabled and IDLESx = 1



The delayed burst entry mode is enabled with DIDLx bit in the HRTIM_OUTxR register (one enable bit per output). It forces a deadtime insertion before the output takes its idle state. Each TIMx output has its own deadtime value:

- DTRx[8:0] on output 1 when DIDL1 = 1
- DTFx[8:0] on output 2 when DIDL2 = 1

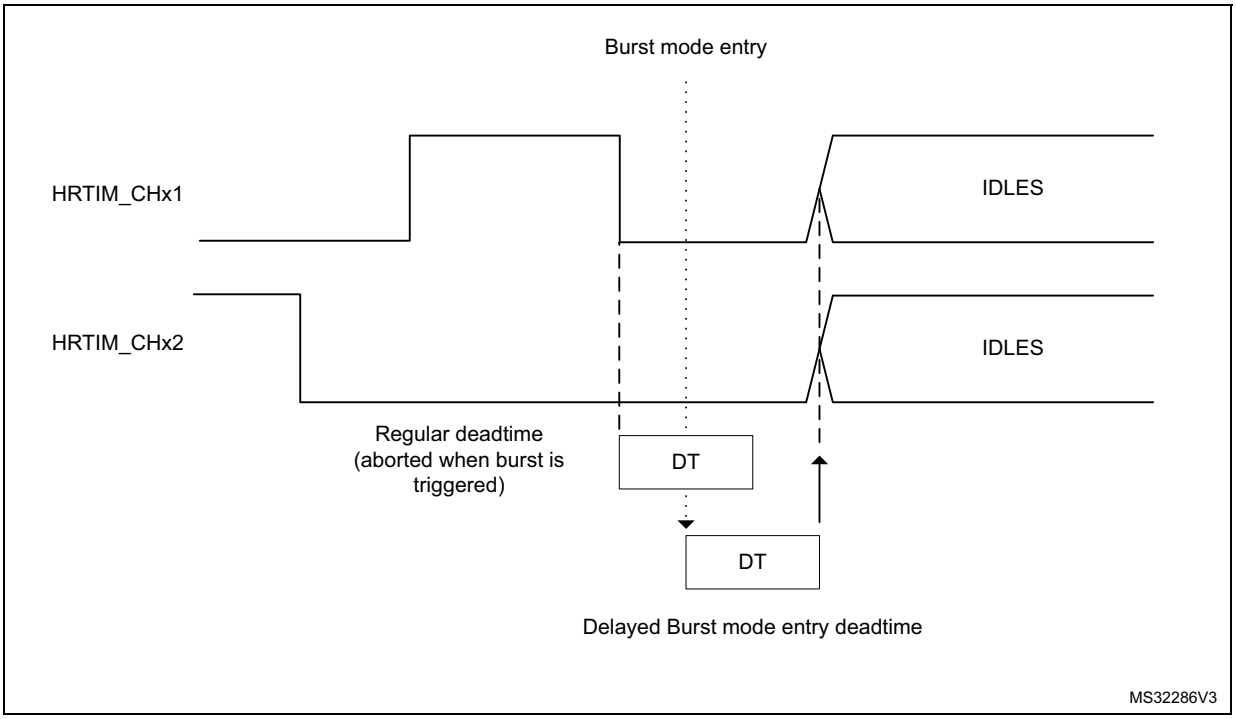
DIDLx bits can be set only if one of the outputs has an active idle level during the burst mode (IDLES = 1) and only when positive deadtimes are used (SDTR/SDTF set to 0).

Note: The delayed burst entry mode uses deadtime generator resources. Consequently, when any of the two DIDLx bits is set and the corresponding timing unit uses the deadtime insertion (DTEN bit set in HRTIM_OUTxR), it is not possible to use the timerx output 2 as a filter for external events (Tx2 filtering signal is not available).

When durations defined by DTRx[8:0] and DTFx[8:0] are lower than three f_{HRTIM} clock cycle periods, the limitations related to the narrow pulse management listed in *Section 40.3.7* must be applied.

When the burst mode entry arrives during the regular deadtime, it is aborted and a new deadtime is restarted corresponding to the inactive period, as on *Figure 426*.

Figure 426. Delayed burst mode entry during deadtime



Burst mode exit

The burst mode exit is either forced by software (in continuous mode) or once the idle period is elapsed (in single-shot mode). In both cases, the counter is re-started immediately (if it was hold in a reset state with MTBM or TxBM bit = 1), but the effective output state transition from the idle to active mode only happens after the programmed set/reset event.

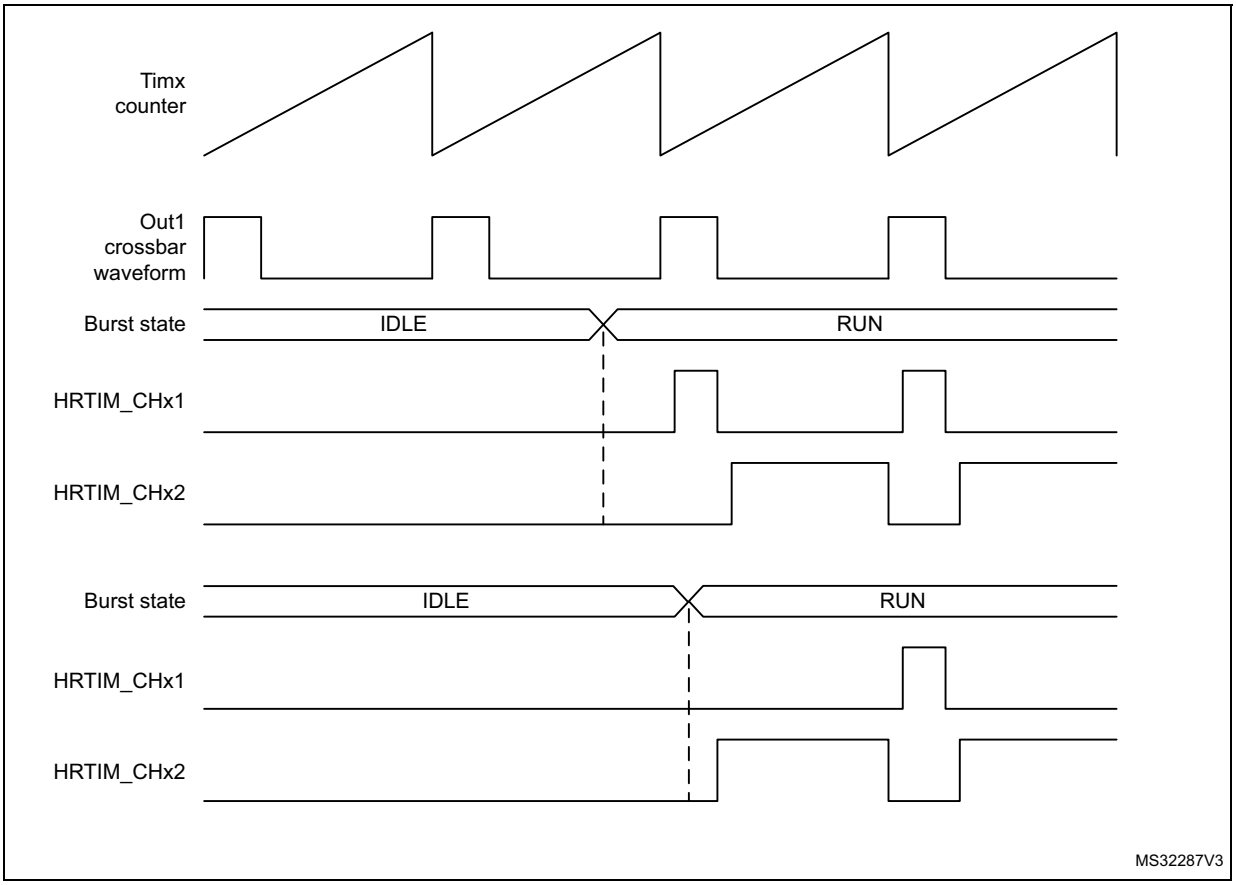
A burst period interrupt is generated in single-shot and continuous modes when BMPERIE enable bit is set in the HRTIM_IER register. This interrupt can be used to synchronize the burst mode exit with a burst period in continuous burst mode.

Figure 427 shows how a normal operation is resumed when the deadtime is enabled. Although the burst mode exit is immediate, this is only effective on the first set event on any of the complementary outputs.

Two different cases are presented:

1. The burst mode ends while the signal is inactive on the crossbar output waveform. The active state is resumed on Tx1 and Tx2 on the set event for the Tx1 output, and the Tx2 output does not take the complementary level on burst exit.
2. The burst mode ends while the crossbar output waveform is active: the activity is resumed on the set event of Tx2 output, and Tx1 does not take the active level immediately on burst exit.

Figure 427. Burst mode exit when the deadtime generator is enabled



The behavior described above is slightly different when the push-pull mode is enabled. The push-pull mode forces an output reset at the beginning of the period if the output is inactive, or symmetrically forces an active level if the output was high during the preceding period.

Consequently, an output with an active idle state can be reset at the time the burst mode is exited even if no transition is explicitly programmed. For symmetrical reasons, an output can be set at the time the burst mode is exited even if no transition is explicitly programmed, in case it was active when it entered in idle state.

Burst mode registers preloading and update

BMPREN bit (burst mode preload enable) is used to have the burst mode compare and period registers preloaded (HRTIM_BMCMP and HRTIM_BMPER).

When BMPREN is set, the transfer from preload to active register happens:

- When the burst mode is enabled (BME = 1),
- At the end of the burst mode period.

A write into the HRTIM_BMPER period register disables the update temporarily, until the HRTIM_BMCMP compare register is written, to ensure the consistency of the two registers when they are modified.

If the compare register only needs to be changed, a single write is necessary. If the period only needs to be changed, it is also necessary to rewrite the compare to have the new values taken into account.

When BMPREN bit is reset, the write access into BMCMPR and BMPER directly updates the active register. In this case, it is necessary to consider when the update is done during the overall burst period, for the two cases below:

a) Compare register update

If the new compare value is above the current burst mode counter value, the new compare is taken into account in the current period.

If the new compare value is below the current burst mode counter value, the new compare is taken into account in the next burst period in continuous mode, and ignored in single-shot mode (no compare match occurs and the idle state lasts until the end of the idle period).

b) Period register update

If the new period value is above the current burst mode counter value, the change is taken into account in the current period.

Note: If the new period value is below the current burst mode counter value, the new period is not taken into account, the burst mode counter overflows (at 0xFFFF) and the change is effective in the next period. In single-shot mode, the counter rolls over at 0xFFFF and the burst mode restarts for another period up to the new programmed value.

Burst mode emulation using a compound register

The burst mode controller only controls one or a set of timers for a single converter. When the burst mode is necessary for multiple independent timers, it is possible to emulate a simple burst mode controller using the DMA and the HRTIM_CMP1CxR compound register, which holds aliases of both the repetition and the compare 1 registers.

This is applicable to a converter which only requires a simple PWM (typically a buck converter), where the duty cycle only needs to be updated. In this case, the CMP1 register is used to reset the output (and define the duty cycle), while it is set on the period event.

In this case, a single 32-bit write access in CMP1CxR is sufficient to define the duty cycle (with the CMP1 value) and the number of periods during which this duty cycle is maintained (with the repetition value). To implement a burst mode, it is then only necessary to transfer by DMA (upon repetition event) two 32-bit data in continuous mode, organized as follows:

CMPC1xR = {REP_Run; CMP1 = Duty_Cycle}, {REP_Idle; CMP1 = 0}

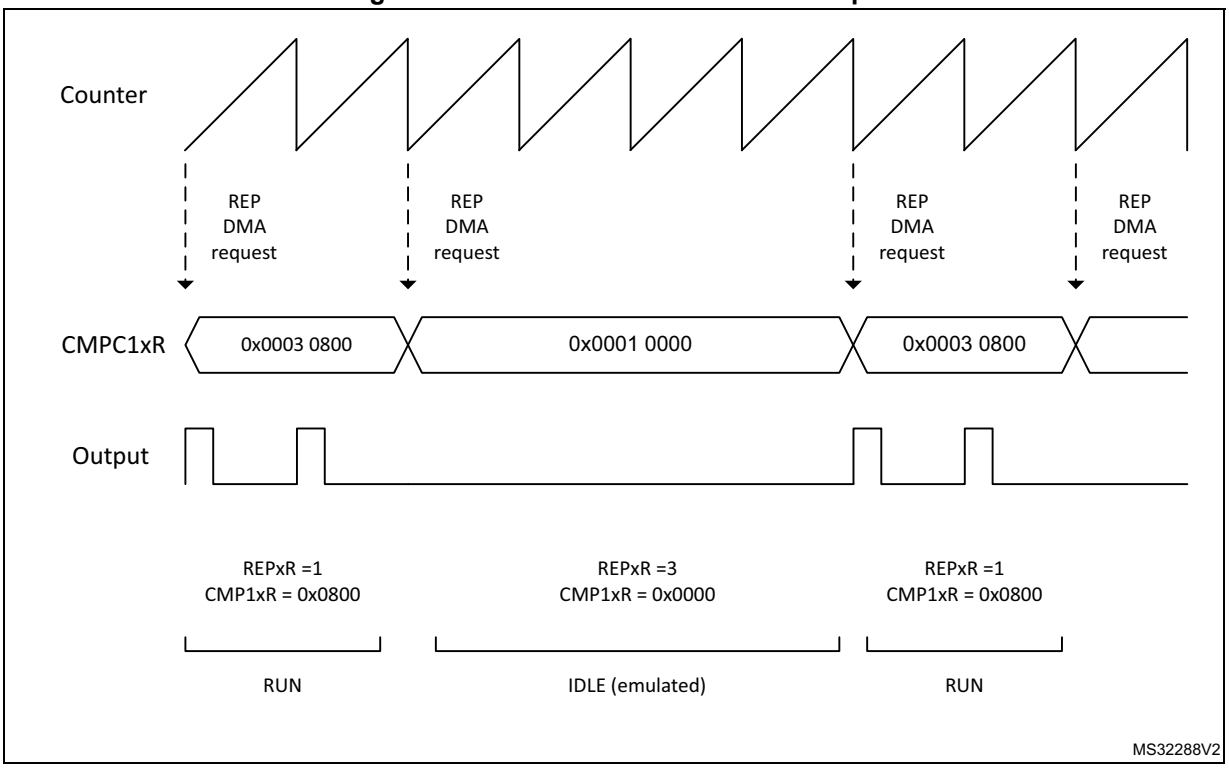
For instance, the values:

{0x0003 0000}: CMP1 = 0 for four periods

{0x0001 0800}: CMP1 = 0x0800 for two period

provide a burst mode with two periods active every six PWM periods, as shown on Figure 428.

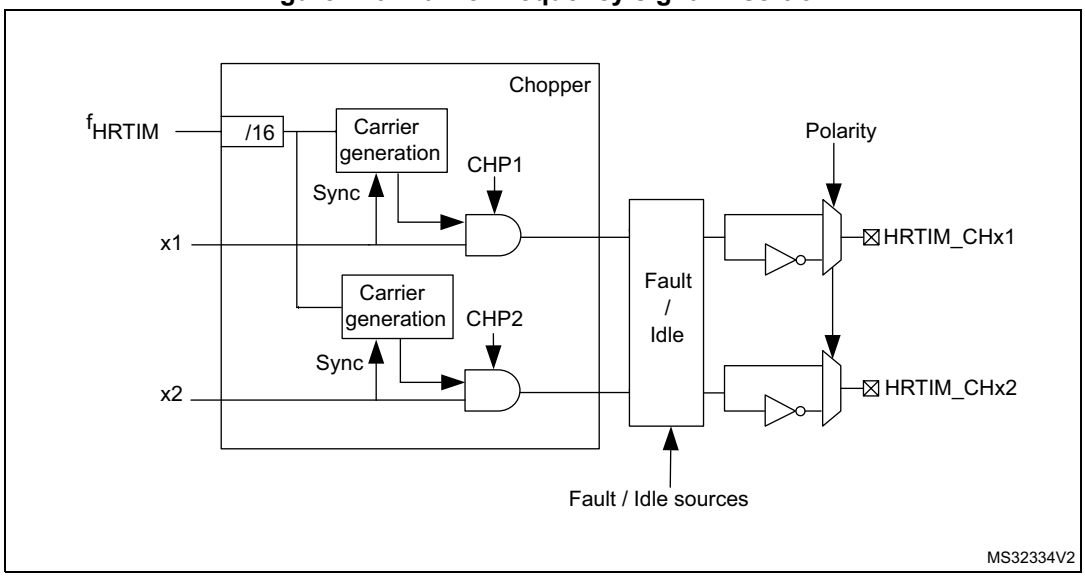
Figure 428. Burst mode emulation example



40.3.16 Chopper

A high-frequency carrier can be added on top of the timing unit output signals to drive isolation transformers. This is done in the output stage before the polarity insertion, as shown on *Figure 429*, using CHP1 and CHP2 bits in the HRTIM_OUTxR register, to enable chopper on outputs 1 and 2, respectively.

Figure 429. Carrier frequency signal insertion



The chopper parameters can be adjusted using the HRIM_CHPxR register, with the possibility to define a specific pulsewidth at the beginning of the pulse, to be followed by a carrier frequency with programmable frequency and duty cycle, as in *Figure 430*.

CARFRQ[3:0] bits define the frequency, ranging from 1.09 MHz to 17.5 MHz (for f_{HRTIM} = 280 MHz) following the formula $F_{CHPFRQ} = f_{HRTIM} / (16 \times (CARFRQ[3:0]+1))$.

The duty cycle can be adjusted by 1/8 step with CARDTY[2:0], from 0/8 up to 7/8 duty cycle. When CARDTY[2:0] = 000 (duty cycle = 0/8), the output waveform only contains the starting pulse following the rising edge of the reference waveform, without any added carrier.

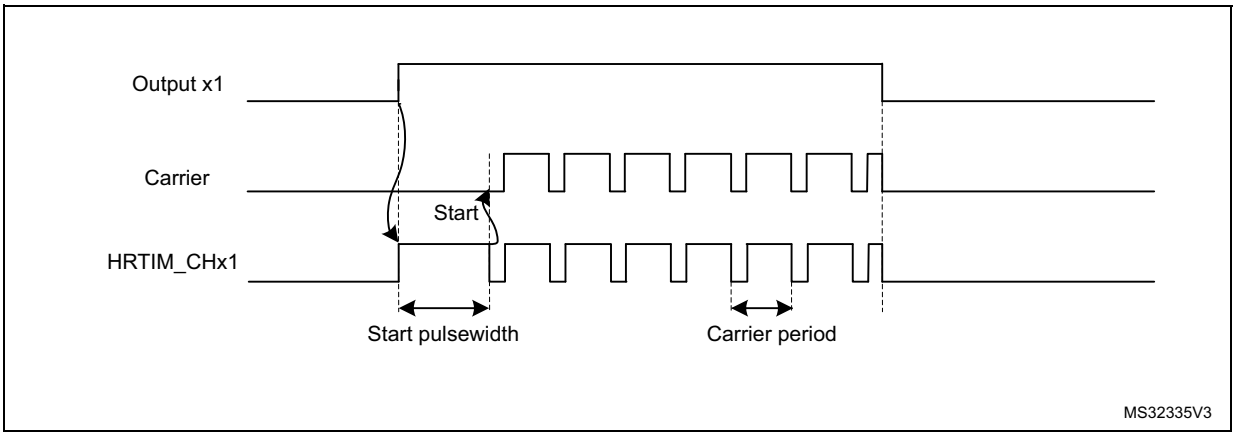
The pulsewidth of the initial pulse is defined using the STRPW[3:0] bitfield as follows: $t_{1STPW} = (STRPW[3:0]+1) \times 16 \times t_{HRTIM}$ and ranges from 57 ns to 910 ns (for f_{HRTIM} = 280 MHz).

The carrier frequency parameters are defined based on the f_{HRTIM} frequency, and do not dependent on the CKPSC[2:0] setting.

In chopper mode, the carrier frequency and the initial pulsewidth are combined with the reference waveform using an AND function. A synchronization is performed at the end of the initial pulse to have a repetitive signal shape.

The chopping signal is stopped at the end of the output waveform active state, without waiting for the current carrier period to be completed. It can thus contain shorter pulses than programmed.

Figure 430. HRTIM outputs with Chopper mode enabled



Note: CHP1 and CHP2 bits must be set prior to the output enable done with TxyOEN bits in the HRTIM_OENR register.

CARFRQ[2:0], CARDTY[2:0], and STRPW[3:0] bitfields cannot be modified while the chopper mode is active (at least one of the two CHPx bits is set).

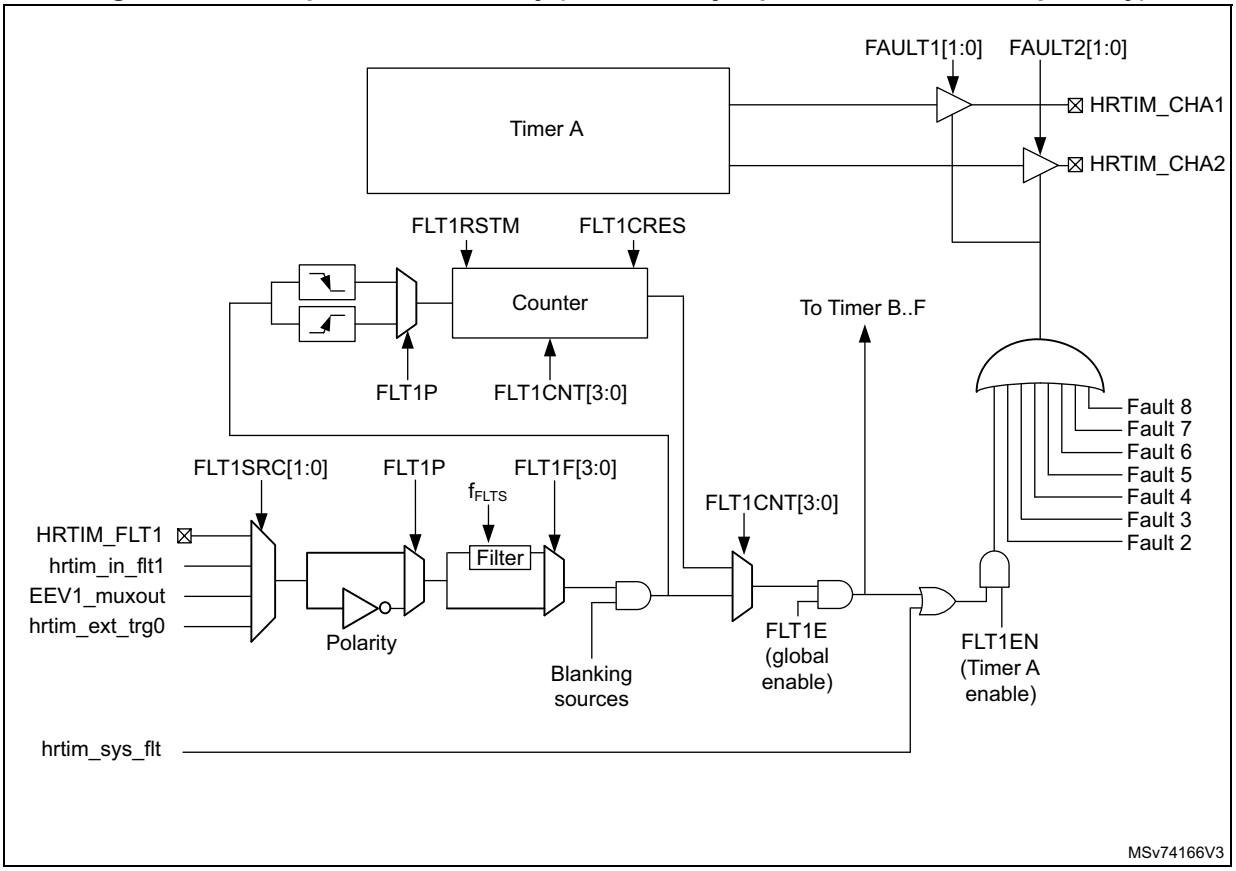
Caution: The chopper mode must only be used with inactive IDLE state (IDLES1 = 0 and IDLES2 = 0).

40.3.17 Fault protection

The HRTIM has a versatile fault protection circuitry to disable the outputs in case of an abnormal operation. Once a fault has been triggered, the outputs take a predefined safe state. This state is maintained until the output is reenabled by software. In case of a permanent fault request, the output remains in its fault state, even if the software attempts to re-enable them, until the fault source disappears.

The HRTIM has eight FAULT input channels; all of them are available and can be combined for each of the eight timing units, as shown on *Figure 431*.

Figure 431. Fault protection circuitry (FAULT1 fully represented, FAULT2..8 partially)



Each fault channel is fully configurable using HRTIM_FLTINR1, HRTIM_FLTINR2 and HRTIM_FLTINR5 registers before being routed to the timing units. FLTxSRC FLTxSRC[1:0] bit selects the source of the fault signal that can be either a digital input or an internal event (built-in comparator output).

Table 416 details the three available sources for each of the six faults channels.

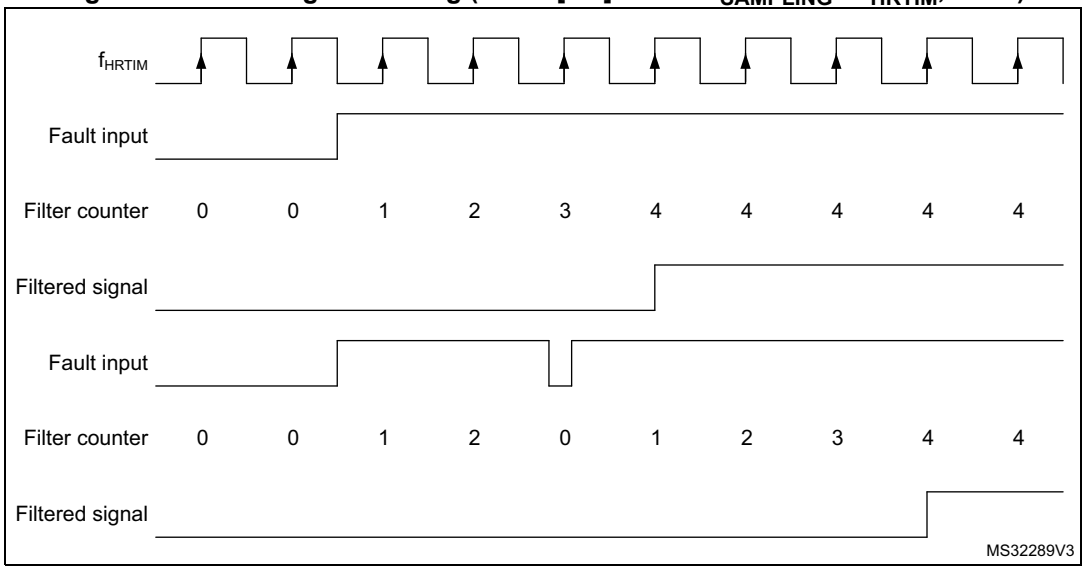
The EEVx_muxout event mentioned in *Figure 431* is taken after the hrtim_eevx[8:1] input multiplexer controlled by the EExSRC[2:0]bits. Refer to *Figure 402* for details.

The polarity of the signal can be selected to define the active level, using the FLTxP polarity bit in HRTIM_FLTINRx registers. If FLTxP = 0, the signal is active at low level; if FLTxP = 1, it is active when high.

The fault information can be filtered after the polarity setting. If FLTxF[3:0] bitfield is set to 0000, the signal is not filtered and acts asynchronously, independently from the f_{HRTIM} clock. For all other FLTxF[3:0] bitfield values, the signal is digitally filtered. The digital filter is made of a counter in which a number N of valid samples are needed to validate a transition on the output. If the input value changes before the counter has reached the value N, the counter is reset and the transition is discarded (considered as a spurious event). If the counter reaches N, the transition is considered as valid and transmitted as a correct external event. Consequently, the digital filter adds a latency to the external events being filtered,

depending on the sampling clock and on the filter length (number of valid samples expected). *Figure 432* shows how a spurious fault signal is filtered.

Figure 432. Fault signal filtering (FLTxF[3:0]= 0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4)



The filtering period ranges from two cycles of the f_{HRTIM} clock up to eight cycles of the f_{FLTS} clock divided by 32. f_{FLTS} is defined using FLTSD[1:0] bits in the HRTIM_FLTINR2 register. *Table 440* summarizes the sampling rate and the filter length. A jitter of 1 sampling clock period must be subtracted from the filter length to take into account the uncertainty due to the sampling and have the effective filtering.

Table 440. Sampling rate and filter length vs FLTFxF[3:0] and clock setting

-	f_{FLTS} vs FLTSD[1:0]				Filter length for f_{HRTIM} = 280 MHz	
FLTFxF[3:0]	**00**	**01**	**10**	**11**	**Min**	**Max**
0001,0010,0011	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}	f_{HRTIM}, N =2 7.1 ns	f_{HRTIM}, N =8 28.6 ns
0100, 0101	f_{HRTIM} /2	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /2, N = 6 43 ns	f_{HRTIM} /16, N = 8 457 ns
0110, 0111	f_{HRTIM} /4	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /4, N = 6 86 ns	f_{HRTIM} /32, N = 8 0.89 µs
1000, 1001	f_{HRTIM} /8	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /8, N = 6 171 ns	f_{HRTIM} /64, N = 8 1.8 µs
1010, 1011, 1100	f_{HRTIM} /16	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /16, N = 5 286 ns	f_{HRTIM} /128, N = 8 3.66 µs
1101, 1110, 1111	f_{HRTIM} /32	f_{HRTIM} /64	f_{HRTIM} /128	f_{HRTIM} /256	f_{HRTIM} /32, N = 5 571 ns	f_{HRTIM} /256, N = 8 7.3 µs

Fault blanking and event counting

The fault inputs can be temporarily disabled to blank spurious fault events. The blanking sources are listed in the *Table 441* below.

Table 441. Fault input blanking events

-	FLTxBLKS = 0, reset-aligned window		FLTxBLKS = 1 moving window	
Fault input	**Blanking window start**	**Blanking window end**	**Blanking window start**	**Blanking window end**
FAULT1	Timer A reset/roll-over	Timer A CMP3 event	Timer A CMP4 event	Timer A CMP3 event
FAULT2	Timer B reset/roll-over	Timer B CMP3 event	Timer B CMP4 event	Timer B CMP3 event
FAULT3	Timer C reset/roll-over	Timer C CMP3 event	Timer C CMP4 event	Timer C CMP3 event
FAULT4	Timer D reset/roll-over	Timer D CMP3 event	Timer D CMP4 event	Timer D CMP3 event
FAULT5	Timer E reset/roll-over	Timer E CMP3 event	Timer E CMP4 event	Timer E CMP3 event
FAULT6	Timer F reset/roll-over	Timer F CMP3 event	Timer F CMP4 event	Timer F CMP3 event
FAULT7	Timer D reset/roll-over	Timer D CMP1 event	Timer D CMP2 event	Timer D CMP1 event
FAULT8	Timer F reset/roll-over	Timer F CMP1 event	Timer F CMP2 event	Timer F CMP1 event

A fault counter is also used to discard multiple spurious fault events and define an acceptance criteria.

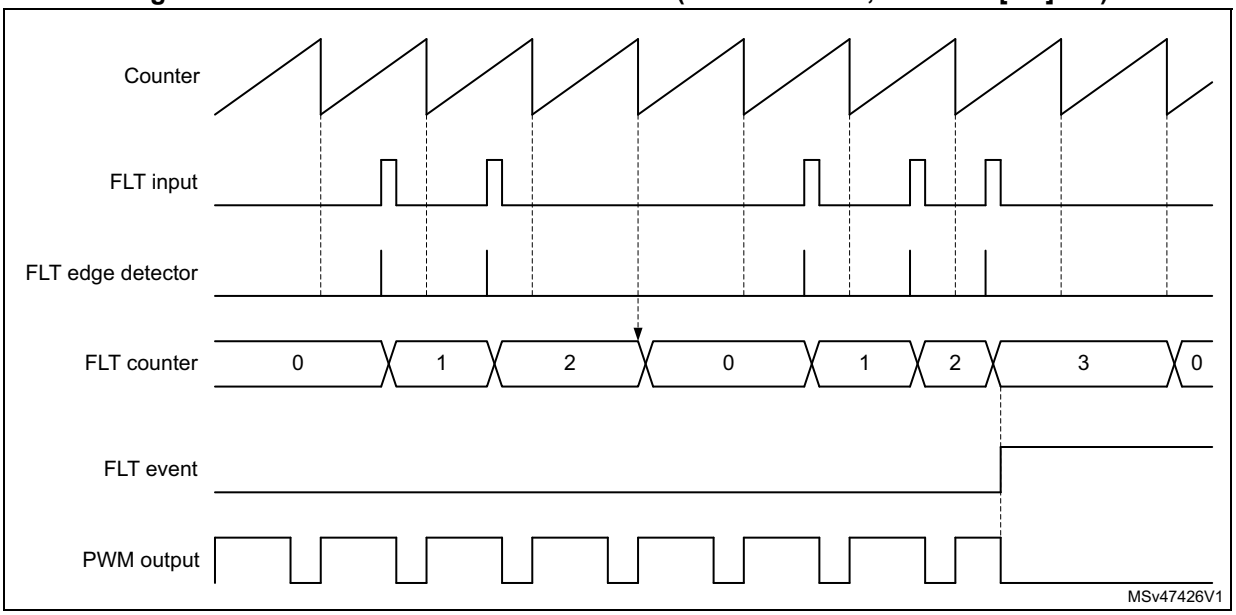
The FLTxCNT[3:0] bitfield selects the FAULTx counter threshold. A fault is considered valid when the number of events is equal to the FLTxCNT[3:0] value.

This FLTxRSTM selects the FAULTx counter reset mode

- 0: fault counter hardware reset on reset/roll-over event, as per the *Table 442* below.
- 1: fault counter reset on each reset/roll-over event only if no event occurs during the last counting period, as shown on *Figure 433* below.

The fault counter can be reset by software with the FLTxCRES bit at any time.

Figure 433. Fault counter cumulative mode (FLTxRSTM = 1, FLTxCNT[3:0] = 2)



A given FLTx input counter can be reset by a single source. The *Table 442* indicates which timer unit is associated with a given fault. This does not prevent having a fault line shared by multiple timers (for example FLT1 with event counter enabled, acting on timer A, timer B, and timer C simultaneously).

Table 442. Fault 1..6 counter reset source

Fault Input	Fault counter reset source
hrtim_flt1[4:1]	Timer A reset/roll-over
hrtim_flt2[4:1]	Timer B reset/roll-over
hrtim_flt3[4:1]	Timer C reset/roll-over
hrtim_flt4[4:1]	Timer D reset/roll-over
hrtim_flt5[4:1]	Timer E reset/roll-over
hrtim_flt6[4:1]	Timer F reset/roll-over
hrtim_flt7[4:1]	Timer D reset/roll-over
hrtim_flt8[4:1]	Timer F reset/roll-over

System fault input (hrtim_sys_flt)

This fault is provided by the MCU Class B circuitry (see *Section 14: System configuration, boot, and security (SBS)* for details) and corresponds to a system fault coming from:

- The clock security system
- The SRAM ECC double error detection
- The Cortex-M33 lockup signal
- The PVD detector
- The DCACHE ECC double error detection
- The flash memory ECC double error detection

This input overrides the FAULT inputs and disables all outputs having FAULTy[1:0] = 01, 10, 11.

For each FAULT channel, a write-once FLTxLCK bit in the HRTIM_FLTxR register is used to lock FLTxE, FLTxP, FLTxSRC, FLTxF[3:0] bits (it renders them read-only), for functional safety purpose. If enabled, the fault conditioning set-up is frozen until the next HRTIM or system reset.

Once the fault signal is conditioned as explained above, it is routed to the timing units. For any of them, the six fault channels are enabled using bits FLT1EN to FLT6EN in the HRTIM_FLTxR register, and they can be selected simultaneously (the sysfault is automatically enabled as long as the output is protected by the fault mechanism). This is used to have, for instance:

- One fault channel simultaneously disabling several timing units
- Multiple fault channels being ORed to disable a single timing unit

A write-once FLTLCK bit in the HRTIM_FLTxR register is used to lock FLTxEN bits (it renders them read-only) until the next reset, for functional safety purpose. If enabled, the timing unit fault-related set-up is frozen until the next HRTIM or system reset.

For each of the timers, the output state during a fault is defined with FAULT1[1:0] and FAULT2[1:0] bits in the HRTIM_OUTxR register (see *Section 40.3.14*).

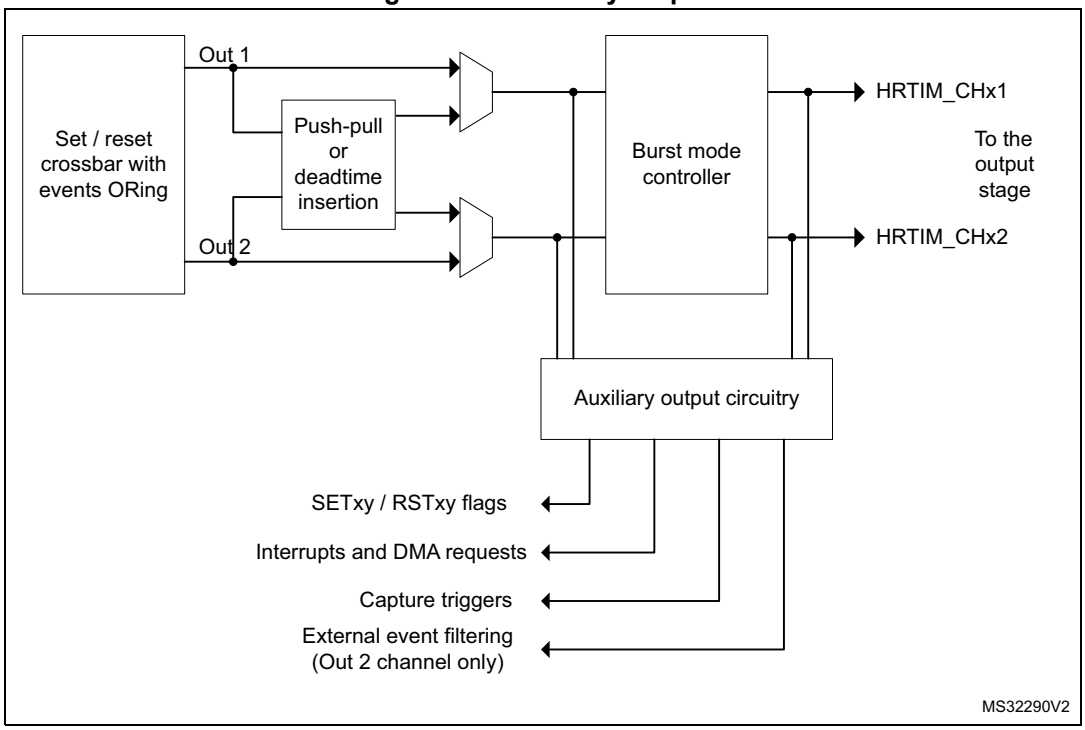
40.3.18 Auxiliary outputs

Timer A to E have auxiliary outputs in parallel with the regular outputs going to the output stage. They provide the following internal status, events, and signals:

- SETxy and RSTxy status flags, together with the corresponding interrupts and DMA requests
- Capture triggers upon output set/reset
- External event filters following a Tx2 output copy (see details in *Section 40.3.9*)

The auxiliary outputs are taken either before or after the burst mode controller, depending on the HRTIM operating mode. An overview is given on *Figure 434*.

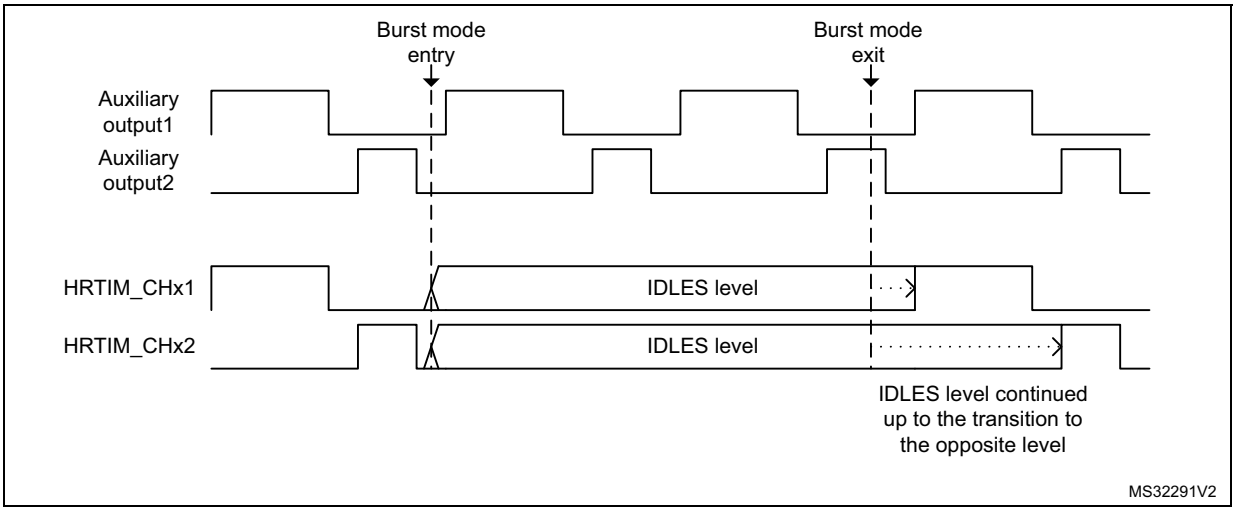
Figure 434. Auxiliary outputs



By default, the auxiliary outputs are copies of outputs Tx1 and Tx2. The exceptions are:

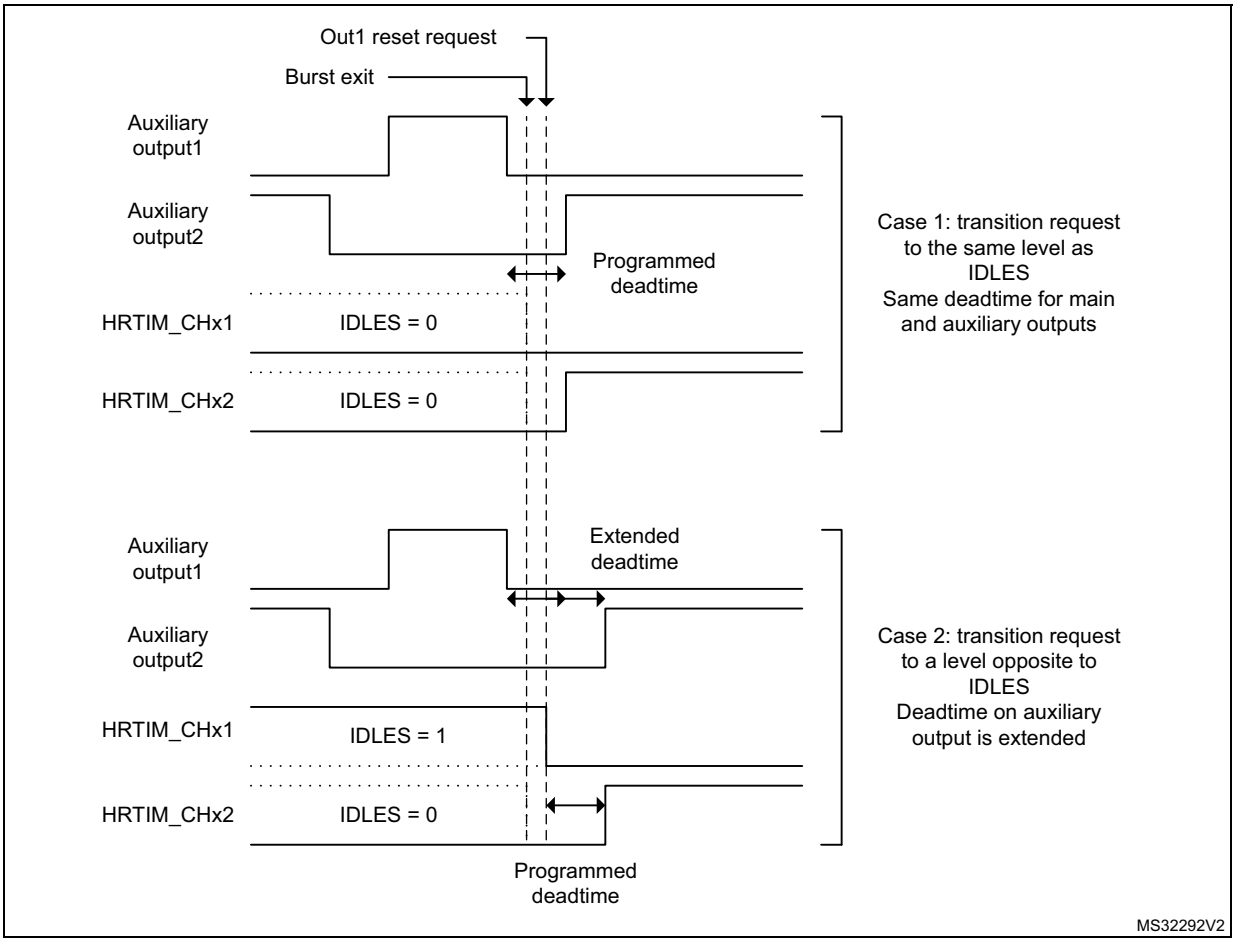
- The delayed idle and the balanced idle protections, when the deadtime is disabled (DTEN = 0). When the protection is triggered, the auxiliary outputs are maintained and follow the signal coming out of the crossbar. On the contrary, if the deadtime is enabled (DTEN = 1), both main and auxiliary outputs are forced to an inactive level.
- The burst mode (TCEN=1, IDLEMx=1); there are two cases:
  a) If DTEN=0 or DIDLx=0, the auxiliary outputs are not affected by the burst mode entry and continue to follow the reference signal coming out of the crossbar (see *Figure 435*).
  b) If the deadtime is enabled (DTEN=1) together with the delayed burst mode entry (DIDLx=1), the auxiliary outputs have the same behavior as the main outputs. They are forced to the IDLES level after a deadtime duration, then they keep this level during all the burst period. When the burst mode is terminated, the IDLES level is maintained until a transition occurs to the opposite level, similarly to the main output.

Figure 435. Auxiliary and main outputs during burst mode (DIDLx = 0)



The signal on the auxiliary output can be slightly distorted when exiting from the burst mode or when reenabling the outputs after a delayed protection, if this happens during a deadtime. In this case, the deadtime applied to the auxiliary outputs is extended so that the deadtime on the main outputs is respected. *Figure 436* gives some examples.

Figure 436. Deadtime distortion on auxiliary output when exiting burst mode



40.3.19 Synchronizing the HRTIM with other timers or HRTIM instances

The HRTIM provides options for synchronizing multiple HRTIM instances, as a master unit (generating a synchronization signal) or as a slave (waiting for a trigger to be synchronized). This feature can also be used to synchronize the HRTIM with other timers, either external or on-chip. The synchronization circuitry is controlled inside the master timer.

Synchronization output

This section explains how the HRTIM must be configured to synchronize external resources and act as a master unit.

Four events can be selected as the source to be sent to the synchronization output. This is done using SYNCSRC[1:0] bits in the HRTIM_MCR register, as follows:

- 00: master timer start
  This event is generated when MCEN bit is set or when the timer is restarted after having reached the period value in single-shot mode. It is also generated on a reset which occurs during the counting (when CONT or RETRIG bits are set).
- 01: master timer compare 1 event
- 10: timer A start
  This event is generated when TACEN bit is set or when the counter is reset and restarts counting in response to this reset. The following counter reset events are not propagated to the synchronization output: counter roll-over in continuous mode, and discarded reset request in single-shot nonretriggerable mode. The reset is only taken into account when it occurs during the counting (CONT or RETRIG bits are set).
- 11: timer A compare 1 event

SYNCOUT[1:0] bits in the HRTIM_MCR register specify how the synchronization event is generated.

The synchronization pulses can be generated on the HRTIM_SCOUT output pin, with SYNCOUT[1:0] = 1x. SYNCOUT[0] bit specifies the polarity of the synchronization signal. If SYNCOUT[0] = 0, the HRTIM_SCOUT pin has a low idle level and issues a positive pulse of 16 f_{HRTIM} clock cycles length for the synchronization). If SYNCOUT[0] = 1, the idle level is high and a negative pulse is generated.

Note: The synchronization pulse is followed by an idle level of 16 f_{HRTIM} *clock cycles during which any new synchronization request is discarded. Consequently, the maximum synchronization frequency is* $f_{HRTIM}/32$.

The idle level on the HRTIM_SCOUT pin is applied as soon as the SYNCOUT[1:0] bits are enabled (i.e. the bitfield value is different from 00).

The synchronization output initialization procedure must be done prior to the configuration of the MCU outputs and counter enable, in the following order:

1. SYNCOUT[1:0] and SYNCSRC[1:0] bitfield configuration in HRTIM_MCR
2. HRTIM_SCOUT pin configuration (see the “General-purpose I/Os” section)
3. Master or timer A counter enable (MCEN or TACEN bit set)

When the synchronization input mode is enabled and starts the counter (using SYNCSTRTM/SYNCSTRTx bits) simultaneously with the synchronization output mode (SYNCSRC[1:0] = 00 or 10), the output pulse is generated only when the counter is starting or is reset while running. Any reset request clearing the counter without causing it to start does not affect the synchronization output.

Synchronization input

The HRTIM can be synchronized by external sources, as per the programming of the SYNCIN[1:0] bits in the HRTIM_MCR register:

- 00: synchronization input is disabled
- 01: reserved configuration
- 10: the On-chip timer TRGO output connected to hrtim_in_sync[2] input (refer to *Table 410* for details).
- 11: a positive pulse on the HRTIM_SCIN input pin (hrtim_in_sync[3])

This bitfield cannot be changed once the destination timer (master timer or timing unit) is enabled (MCEN and/or TxCEN bit set).

The HRTIM_SCIN input is rising-edge sensitive. The timer behavior is defined with the following bits present in HRTIM_MCR and HRTIM_TIMxCR registers (see *Table 443* for details):

- Synchronous start: the incoming signal starts the timer's counter (SYNCSTRTM and/or SYNCSTRTx bits set). TxCEN (MCEN) bits must be set to have the timer enabled and the counter ready to start. In continuous mode, the counter does not start until the synchronization signal is received.
- Synchronous reset: the incoming signal resets the counter (SYNCRSTM and/or SYNCRSTx bits set). This event decrements the repetition counter as any other reset event.

The synchronization events are taken into account only once the related counters are enabled (MCEN or TxCEN bit set). A synchronization request triggers a SYNC interrupt.

Note: A synchronized start event resets the counter if the current counter value is above the active period value.

The effect of the synchronization event depends on the timer operating mode, as summarized in *Table 443*.

Table 443. Effect of sync event versus timer operating modes

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot non-retriggerable	0	1	Start events are taken into account when the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be re-started with the synchronization). Any reset occurring during the counting is ignored (as during regular non-retriggerable mode).
	1	X	Reset events are starting the timer counting. They are taken into account only if the counter is stopped and: – once the MCEN or TxCEN bits are set – once the period has been reached. When multiple reset requests are selected (from HRTIM_SCIN and from internal events), only the first arriving request is taken into account.

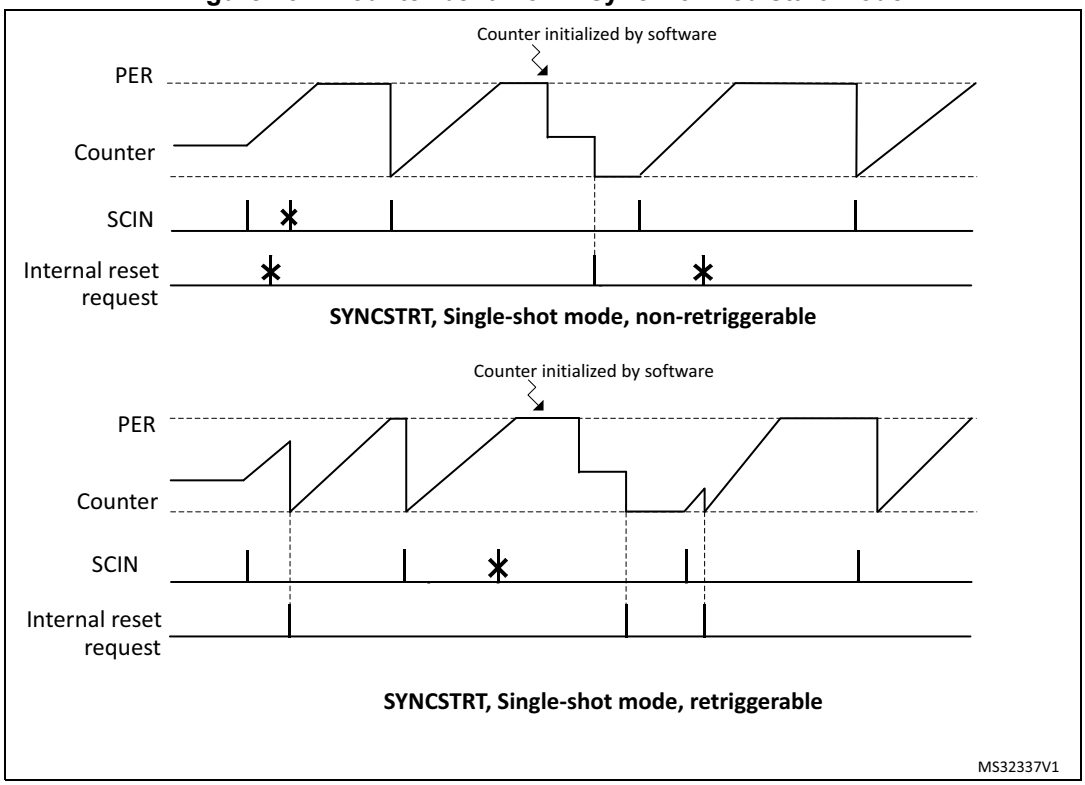
Table 443. Effect of sync event versus timer operating modes (continued)

Operating mode	SYNC RSTx	SYNC STRTx	Behavior following a SYNC reset or start event
Single-shot retriggerable	0	1	The counter start is effective only if the counter is not started or period is elapsed. Any synchronization event occurring after counter start has no effect. A start occurring when the counter is stopped at the period value resets the counter. A reset request clears the counter but does not start it (the counter can solely be started by the synchronization). A reset occurring during counting is taken into account (as during regular retriggerable mode).
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM's timer counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.
Continuous mode	0	1	The timer is enabled (MCEN or TxCEN bit set) and is waiting for the synchronization event to start the counter. Any synchronization event occurring after the counter start has no effect (the counter can solely be started by the synchronization). A reset request clears the counter but does not start it.
	1	X	The reset from HRTIM_SCIN is taken into account as any HRTIM's timer counter reset from internal events and is starting or re-starting the timer counting. When multiple reset requests are selected, the first arriving request is taken into account.

When a synchronization reset event occurs within the same f_{HRTIM} clock cycle as the period event, this reset is postponed to a programmed period event (since both events are causing a counter roll-over). This applies only when the high-resolution is active (CKPSC[2:0] < 5).

Figure 437 presents how the synchronized start is done in single-shot mode.

Figure 437. Counter behavior in synchronized start mode



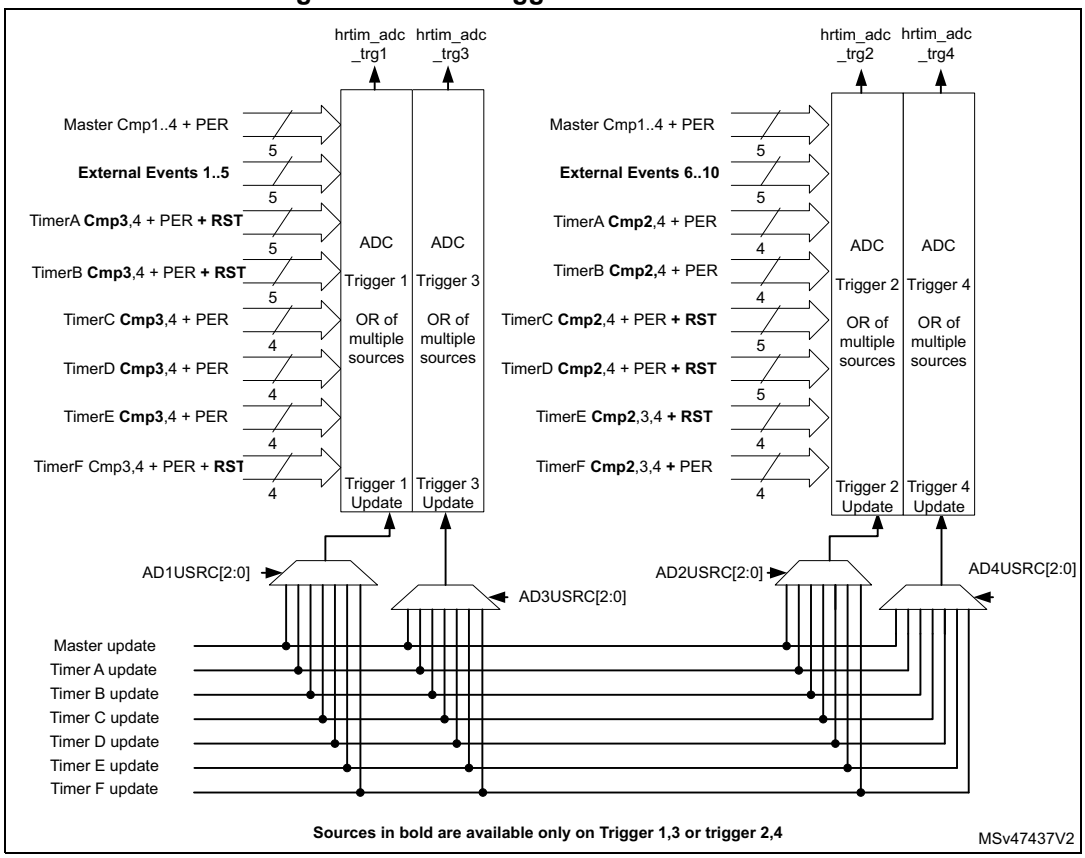
40.3.20 ADC triggers

The ADCs can be triggered by the master and the six timing units.

10 independent triggers are available for both the regular and the injected sequencers of the ADCs. The external events can be used as triggers. They are taken right after the conditioning defined in the HRTIM_EECRx registers, and are not depending on the EEFxR1 and EEFxR2 register settings.

Up to 32 events can be combined (ORed) for ADC triggers 1 to 4, in HRTIM_ADC1R to HRTIM_ADC4R registers, as shown on *Figure 438*. The ADC triggers 1/3 and 2/4 are using the same source set. A multiple triggering is possible within a single switching period by selecting several sources simultaneously. A typical use case is for a nonoverlapping multiphase converter, where all phases can be sampled in a row using a single ADC trigger output.

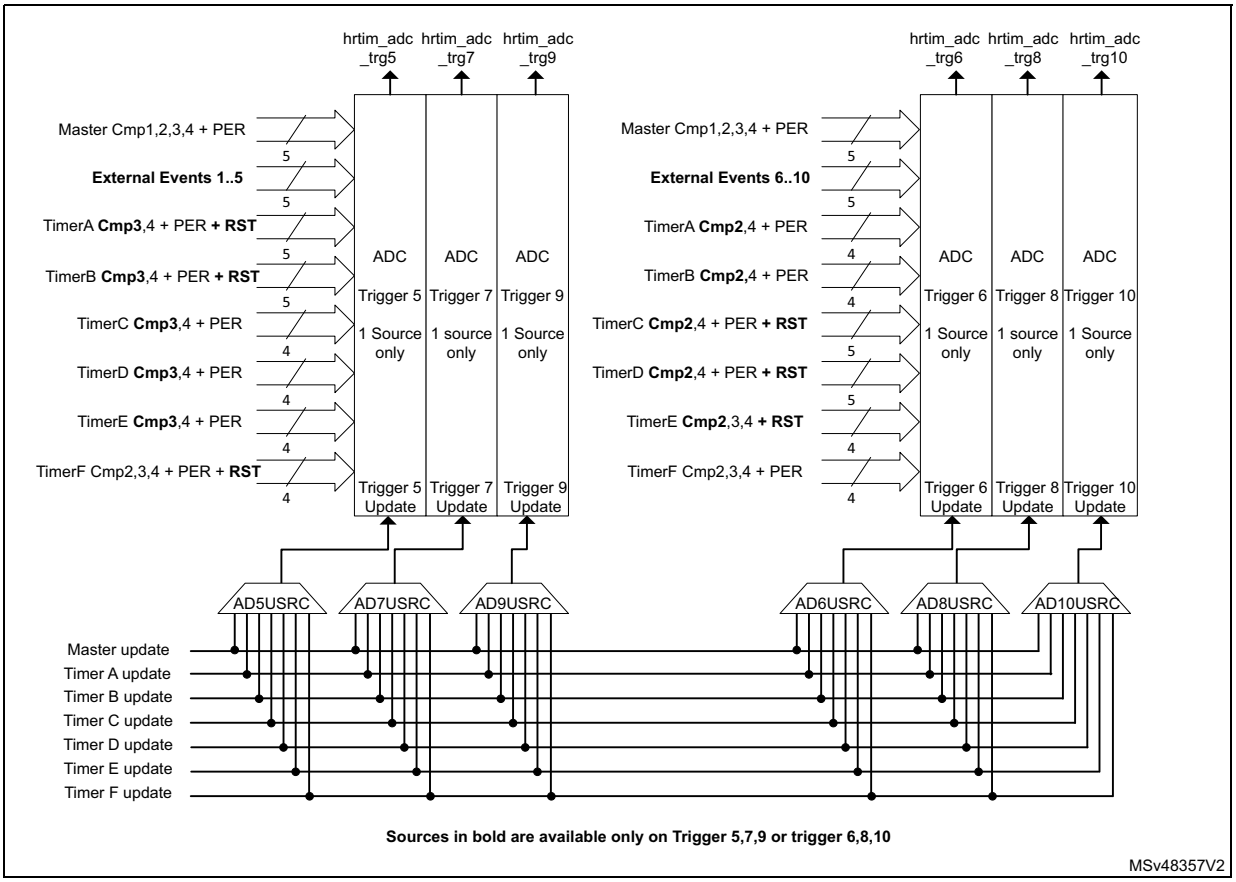
Figure 438. ADC trigger selection overview



The ADC triggers 5 to 10 are configured in the HRTIM_ADCER register, as shown on *Figure 439*. The ADC triggers 5/7/9 and 6/8/10 are using the same source set.

A single source can be selected at once for these triggers (1 out of 32 possible events).

Figure 439. ADC triggers



HRTIM_ADC1R to HRTIM_ADC4R and HRTIM_ADCER registers are preloaded and can be updated synchronously with the timer they are related to. The update sources are defined with ADxUSRC[2:0] bits in the HRTIM_CR1 and HRTIM_ADCUR registers.

For instance, if ADC trigger 1 outputs timer A CMP2 events (HRTIM_ADC1R = 0x0000 0400), HRTIM_ADC1R is typically updated simultaneously with timer A (AD1USRC[2:0] = 001).

When the preload is disabled (PREEN bit reset) in the source timer, the HRTIM_ADCxR registers are not preloaded either: a write access results in an immediate update of the trigger source.

ADC postscaler

A postscaling unit is used to reduce the ADC trigger rate with respect to the switching frequency.

Each ADC trigger rate can be individually adjusted using the ADCxPSC[4:0] bits in the HRTIM_ADCxPS1 and HRTIM_ADCxPS2 registers as shown in *Figure 440* below for the up-counting mode.

In the center-aligned mode, the ADC postscaling is done on events selected with the ADROM[1:0] bitfield, programmed in the source timer, as shown in *Figure 441*. The

ADROM[1:0] bitfield is coding for any event that can trigger the ADC: reset, roll-over (period) and compare event:

- ADROM[1:0] = 00: event generated both during up and down-counting phases
- ADROM[1:0] = 01: event generated during down-counting phases
- ADROM[1:0] = 10: event generated during up-counting phases

The ADC postscaler programming registers are preloaded and can be updated on-the-fly without stopping the timers.

Figure 440. ADC trigger post-scaling in up-counting mode

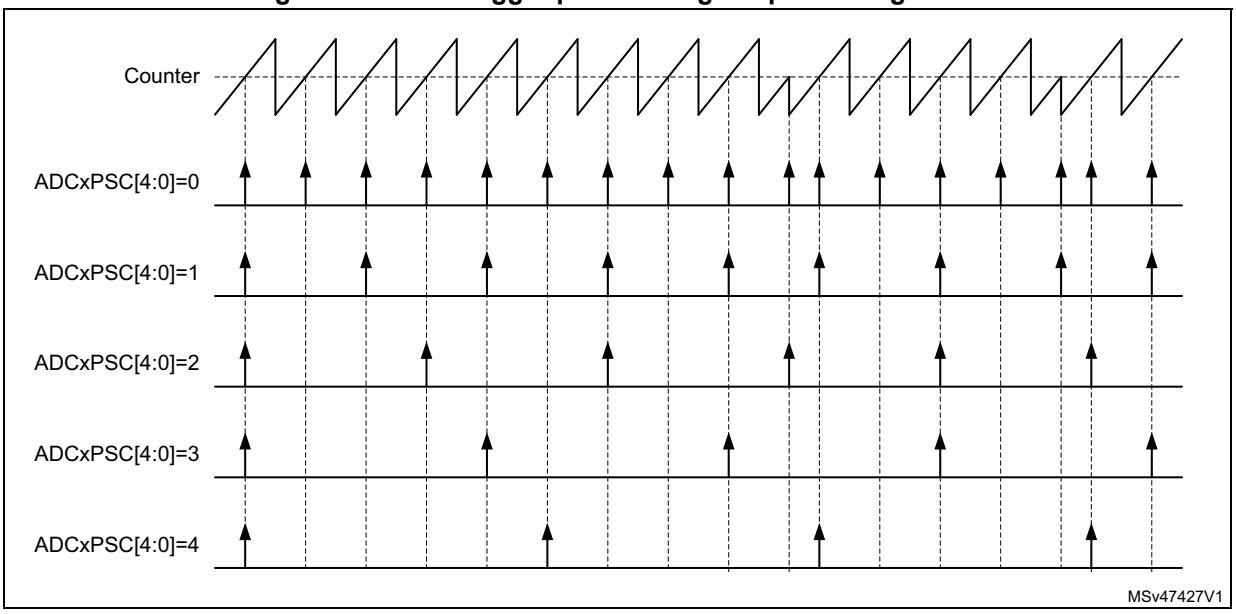
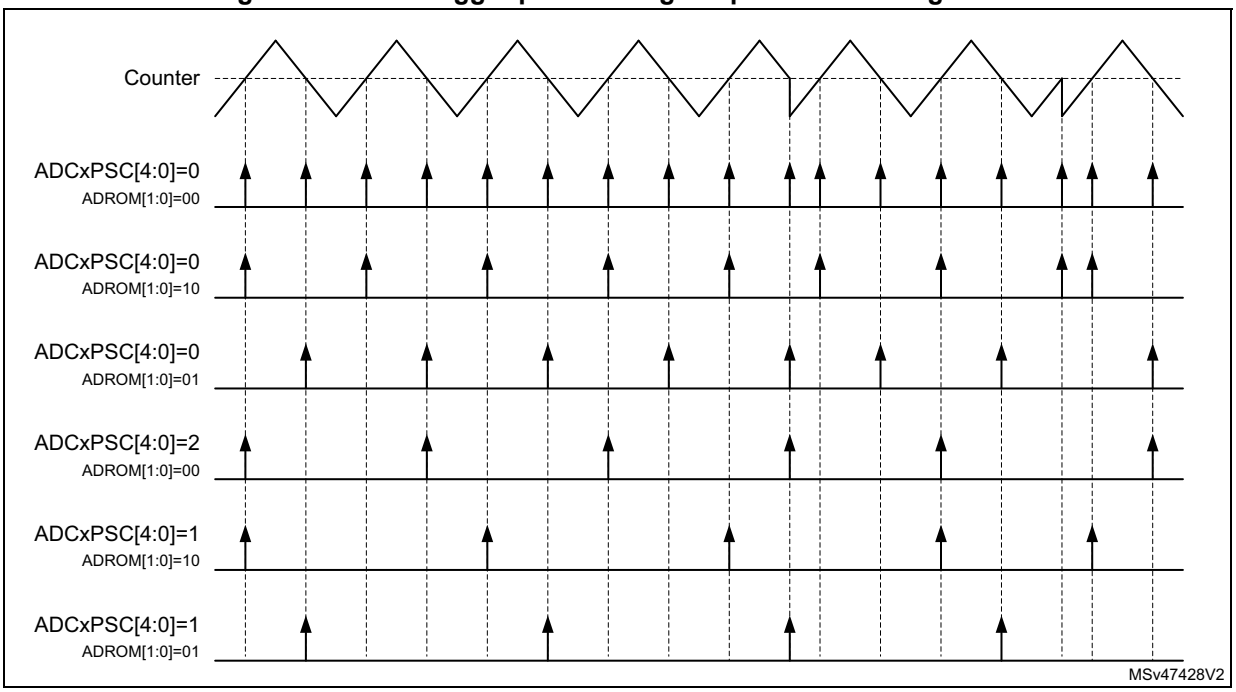


Figure 441. ADC trigger post-scaling in up/down counting mode



40.3.21 DAC triggers

The HRTIM is used to have the embedded DACs updated synchronously with the timer updates.

The update events from the master timer and the timer units can generate DAC update triggers on any of the three hrtim_dac_trgx outputs.

Note: Each timer has its own DAC-related control register.

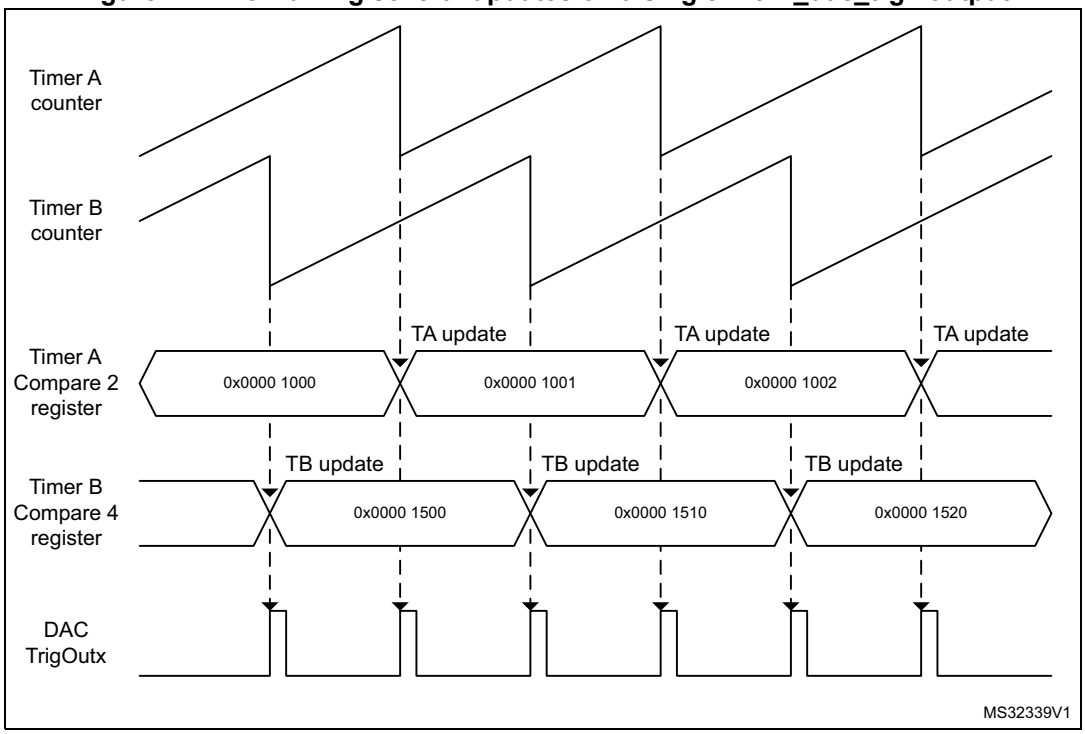
DACSYNC[1:0] bits of the HRTIM_MCR and HRTIM_TIMxCR registers are programmed as follows:

- 00: No update generated
- 01: Update generated on hrtim_dac_trg1
- 10: Update generated on hrtim_dac_trg2
- 11: Update generated on hrtim_dac_trg3

An output pulse of one f_{HRTIM} clock period is generated on the hrtim_dac_trgx output.

When DACSYNC[1:0] bits are enabled in multiple timers, the hrtim_dac_trgx output consists of an OR of all timers' update events. For instance, if DACSYNC = 1 in timer A and in timer B, the update event in timer A is ORed with the update event in timer B to generate a DAC update trigger on the corresponding hrtim_dac_trgx output, as shown on *Figure 442*.

Figure 442. Combining several updates on a single hrtim_dac_trgx output



Refer to *Table 418: HRTIM DAC triggers connections* for connections to the DACs.

Dual channel DAC trigger

Slope compensation techniques and hysteretic control to be easily implemented using HRTIM built-in features and the DAC sawtooth generator. The principle is to have a DAC generating a decreasing saw-tooth synchronized with the PWM period, or a square wave synchronized with PWM signal.

This mode is enabled with the DCDE bit in the TIMxCR2 register. This bit cannot be changed once the timer is operating (TxEN bit set).

It uses two trigger outputs, as shown on the *Figure 443* below:

- The hrtim_dac_reset_trgx generates DAC reset/update events
- The hrtim_dac_step_trgx generates requests for incremental DAC value changes

The DCDR bit in the TIMxCR2 register defines when the hrtim_dac_reset_trgx trigger is generated:

- DCDR = 0: the trigger is generated on counter reset or roll-over event
- DCDR = 1: the trigger is generated on output 1 set event

Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

The DCDS bit in the TIMxCR2 register defines when the hrtim_dac_step_trgx trigger is generated:

- DCDS = 0: the trigger is generated on compare 2 event
- DCDS = 1: the trigger is generated on output 1 reset event

The DCDR and DCDS bits allows the following use cases to be covered:

- Edge-aligned slope compensation (DCDR = DCDS = 0): the DAC's sawtooth starts on PWM period beginning and multiple triggers are generated during the period
- Center-aligned slope compensation (DCDR = 1 DCDS = 0): the DAC's sawtooth starts on the output set event and multiple triggers are generated during the period
- Hysteretic controller: the DAC value must be changed twice per period, when the output state changes. 2 triggers are generated per PWM period. In edge-aligned mode (DCDR=0, DCDS =1), the triggers are generated on counter reset or roll-over. In center-aligned mode (DCDR=1, DCDS=1), the triggers are generated when the output is set.

The compare 2 has a particular operating mode when the DCDE is set and the DCDS bit is reset. The active comparison value is automatically updated as soon as a compare match has occurred, so that the trigger can be repeated periodically with a period equal to the CMP2 value, as represented on *Figure 443*.
The dual channel DAC trigger with DCDS bit reset (compare 2 event used) must not be used simultaneously with modes using CMP2 (triple/quad interleaved and triggered-half modes).

Note: The CMP2 value can be changed on-the-fly. The new value is taken into account on the next coming compare match.

Note: When the DCDS bit is reset, the CMP2 value must not be modified by other mechanisms: the interleaved, triggered half and balanced idle modes must be disabled.

Table 444 below gives an example, for generating six triggers within a PWM period. It shows that it is necessary to round up the division result to the upper value.

Let's consider a counter period TIMxPER = 8192. Dividing 8192 by 6 yields 1365.33.

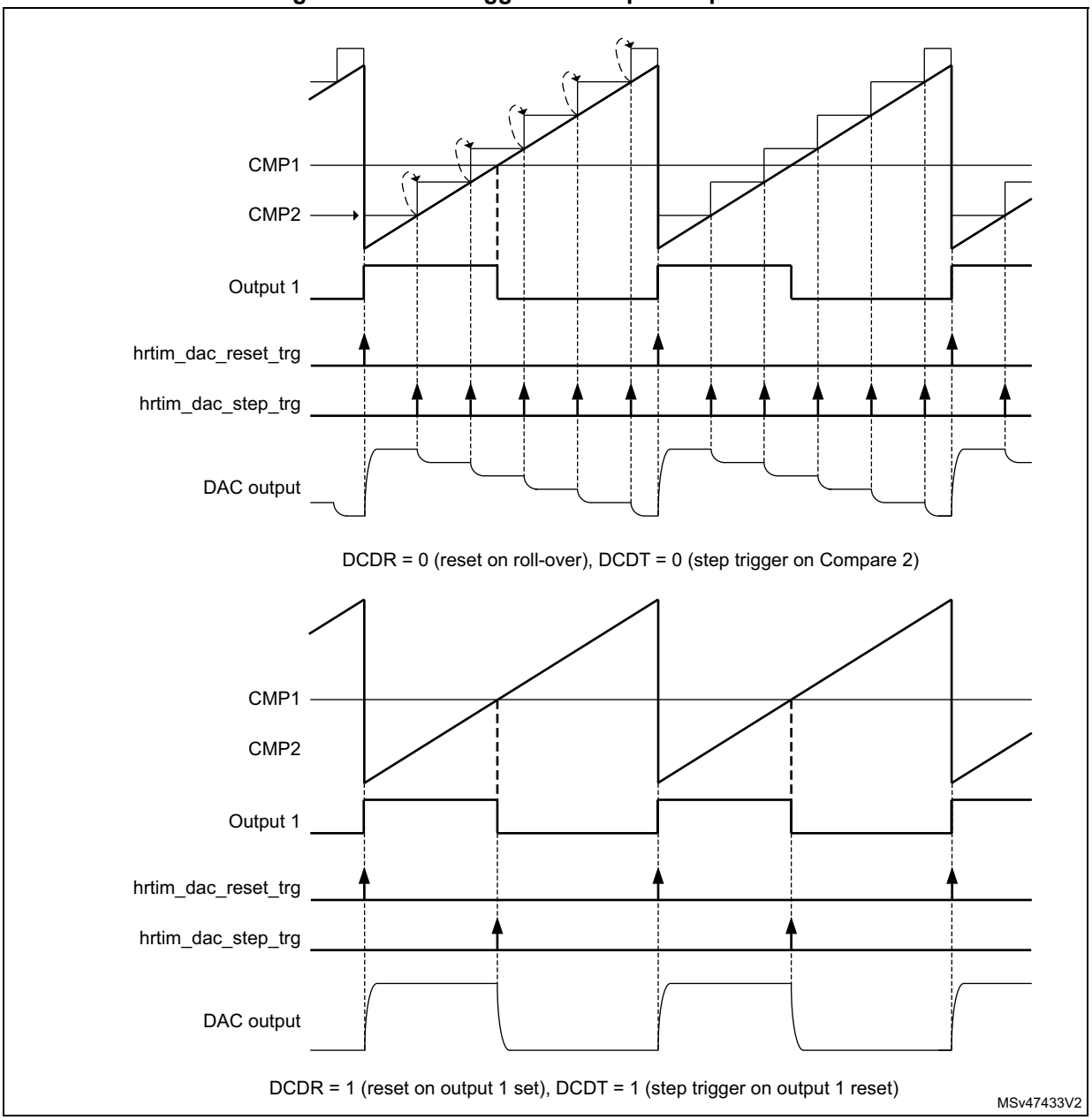
- Round down value: 1365: 7 triggers are generated, the 6th and 7th being very close (respectively for counter = 8190 and 8192)
- Round up value:1366: six triggers are generated. The sixth trigger on hrtim_dac_step_trg (for counter = 8192) is aborted by the counter roll-over from 8192 to 0.

Table 444. DAC dual channel trigger example

-	CMP2 = 1365	hrtim_dac_ reset_trg	hrtim_dac_ step_trg	CMP2 = 1366	hrtim_dac_ reset_trg	hrtim_dac_ step_trg
Counter value	1365	-	1	1366	-	1
	2730	-	2	2732	-	2
	4095	-	3	4098	-	3
	5460	-	4	5464	-	4
	6825	-	5	6830	-	5
	8190	-	6	8192	6	-
	8192	7	-	1366	-	1
	1365	-	1	2732	-	2
	...	-	-	...	-	-

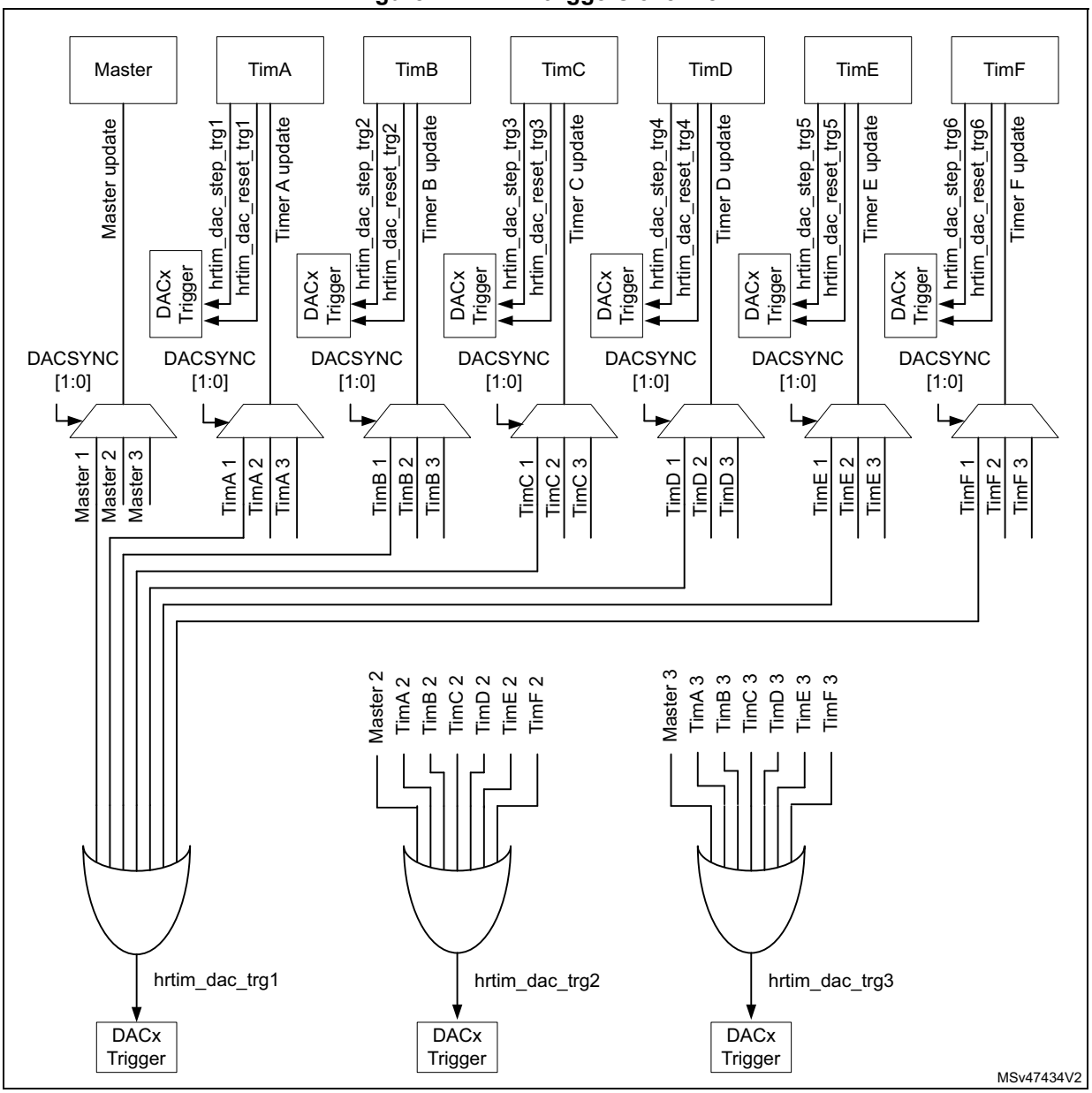
Note: In centered-pattern mode, it is mandatory to have an even number of triggers per switching period, so as to avoid unevenly spaced triggers around counter's peak value.

Figure 443. DAC triggers for slope compensation



The *Figure 444* below provides an overview of all the available DAC triggers.

Figure 444. DAC triggers overview



40.3.22 Interrupts

7 interrupts can be generated by the master timer:

- Master timer registers update
- Synchronization event received
- Master timer repetition event
- Master compare 1 to 4 event

14 interrupts can be generated by each timing unit:

- Crossbar override rearm delay elapsed
- Delayed protection triggered
- Counter reset or roll-over event
- Output 1 and output 2 reset (transition active to inactive)
- Output 1 and output 2 set (transition inactive to active)
- Capture 1 and 2 events
- Timing unit registers update
- Crossbar override trigger
- Repetition event
- Compare 1 to 4 event

8 global interrupts are generated for the whole HRTIM:

- System fault and fault 1 to 6 (regardless of the timing unit attribution)
- DLL calibration done
- Burst mode period completed

The interrupt requests are grouped in eight vectors as follows:

- hrtim_it1: master timer interrupts (master update, sync input, repetition, MCMP1..4) and global interrupt except faults (burst mode period and DLL ready interrupts)
- hrtim_it2: TIMA interrupts
- hrtim_it3: TIMB interrupts
- hrtim_it4: TIMC interrupts
- hrtim_it5: TIMD interrupts
- hrtim_it6: TIME interrupts
- hrtim_it7: TIMF interrupts
- hrtim_it8: Dedicated vector all fault interrupts to allow high-priority interrupt handling

Table 445 is a summary of the interrupt requests, their mapping and associated control, and status bits.

Table 445. HRTIM interrupt summary

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_it1	Burst mode period completed	BMPER	BMPERIE	BMPERC
	DLL calibration done	DLLRDY	DLLRDYIE	DLLRDYC
	Master timer registers update	MUPD	MUPDIE	MUPDC
	Synchronization event received	SYNC	SYNCIE	SYNCC
	Master timer repetition event	MREP	MREPIE	MREPC
	Master compare 1 to 4 event	MCMP1	MCMP1IE	MCP1C
		MCMP2	MCMP2IE	MCP2C
		MCMP3	MCMP3IE	MCP3C
		MCMP4	MCMP4IE	MCP4C

Table 445. HRTIM interrupt summary (continued)

Interrupt vector	Interrupt event	Event flag	Enable control bit	Flag clearing bit
hrtim_it2 hrtim_it3 hrtim_it4 hrtim_it5 hrtim_it6 hrtim_it7	Crossbar override rearm delay elapsed	CORDE	CORDEIE	CORDEC
	Delayed protection triggered	DLYPRT	DLYPRTIE	DLYPRTC
	Counter reset or roll-over event	RST	RSTIE	RSTC
	Output 1 and output 2 reset (transition active to inactive)	RSTx1	RSTx1IE	RSTx1C
		RSTx2	RSTx2IE	RSTx2C
	Output 1 and output 2 set (transition inactive to active)	SETx1	SETx1IE	SETx1C
		SETx2	SETx2IE	SETx2C
	Capture 1 and 2 events	CPT1	CPT1IE	CPT1C
		CPT2	CPT2IE	CPT2C
	Timing unit registers update	UPD	UPDIE	UPDC
	Crossbar override trigger	COT	COTIE	COTC
	Repetition event	REP	REPIE	REPC
	Compare 1 to 4 event	CMP1	CMP1IE	CMP1C
		CMP2	CMP2IE	CMP2C
		CMP3	CMP3IE	CMP3C
		CMP4	CMP4IE	CMP4C
hrtim_it8	System fault	SYSFLT	SYSFLTIE	SYSFLTC
	Fault 1 to 6	FLT1	FLT1IE	FLT1C
		FLT2	FLT2IE	FLT2C
		FLT3	FLT3IE	FLT3C
		FLT4	FLT4IE	FLT4C
		FLT5	FLT5IE	FLT5C
		FLT6	FLT6IE	FLT6C

40.3.23 DMA

Most of the events able to generate an interrupt can also generate a DMA request, even both simultaneously. Each timer (master, TIMA...F) has its own DMA enable register.

The individual DMA requests are ORed into seven channels as follows:

- 1 channel for the master timer
- 1 channel per timing unit (TIMA...F)

Note: Before disabling a DMA channel (DMA enable bit reset in TIMxDIER), it is necessary to disable first the DMA controller.

Table 446 is a summary of the events with their associated DMA enable bits.

Table 446. HRTIM DMA request summary

DMA Channel	Event	DMA capable	DMA enable bit
hrtim_dma1 (master timer)	Burst mode period completed	No	N/A
	DLL calibration done	No	N/A
	Master timer registers update	Yes	MUPDDE
	Synchronization event received	Yes	SYNCDE
	Master timer repetition event	Yes	MREPDE
	Master compare 1 to 4 event	Yes	MCMP1DE
		Yes	MCMP2DE
		Yes	MCMP3DE
		Yes	MCMP4DE
hrtim_dma2 (timer A) hrtim_dma3 (timer B) hrtim_dma4 (timer C) hrtim_dma5 (timer D) hrtim_dma6 (timer E) hrtim_dma7 (timer F)	Crossbar override rearm delay elapsed	Yes	CORDEDE
	Delayed protection triggered	Yes	DLYPRTDE
	Counter reset or roll-over event	Yes	RSTDE
	Output 1 and output 2 reset (transition active to inactive)	Yes	RSTx1DE
		Yes	RSTx2DE
	Output 1 and output 2 set (transition inactive to active)	Yes	SETx1DE
		Yes	SETx2DE
	Capture 1 and 2 events	Yes	CPT1DE
		Yes	CPT2DE
	Timing unit registers update	Yes	UPDDE
	Crossbar override trigger	Yes	COTDE
	Repetition event	Yes	REPDE
	Compare 1 to 4 event	Yes	CMP1DE
		Yes	CMP2DE
		Yes	CMP3DE
		Yes	CMP4DE
N/A	System fault	No	N/A
	Fault 1 to 6	No	N/A
	Burst mode period completed	No	N/A
	DLL calibration done	No	N/A

Burst DMA transfers

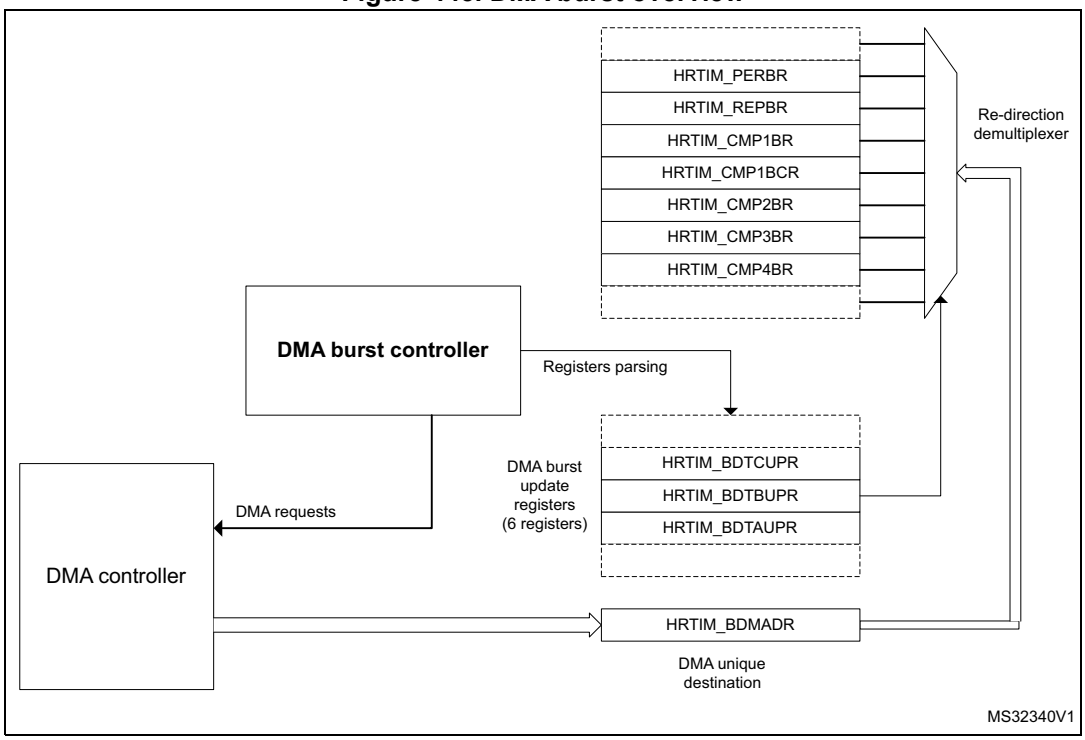
In addition to the standard DMA requests, the HRTIM features a DMA burst controller to have multiple registers updated with a single DMA request. This allows to:

- Update multiple data registers with one DMA channel only,
- Reprogram dynamically one or several timing units, for converters using multiple timer outputs.

The burst DMA feature is only available for one DMA channel, but any of the six channels can be selected for burst DMA transfers.

The principle is to program which registers are to be written by DMA. The master timer and TIMA..E have the burst DMA update register, where most of their control and data registers are associated with a selection bit: HRTIM_BDMUPR, HRTIM_BDTAUPR to HRTIM_BDTEUPR (this is applicable only for registers with write accesses). A redirection mechanism is used to forward the DMA write accesses to the HRTIM registers automatically, as shown on *Figure 445*.

Figure 445. DMA burst overview



When the DMA trigger occurs, the HRTIM generates multiple 32-bit DMA requests and parses the update register. If the control bit is set, the write access is redirected to the associated register. If the bit is reset, the register update is skipped and the register parsing is resumed until a new bit set is detected, to trigger a new request. Once the six update registers (HRTIM_BDMUPR, 5x HRTIM_BDTxUPR) are parsed, the burst is completed and the system is ready for another DMA trigger (see the flowchart on *Figure 446*).

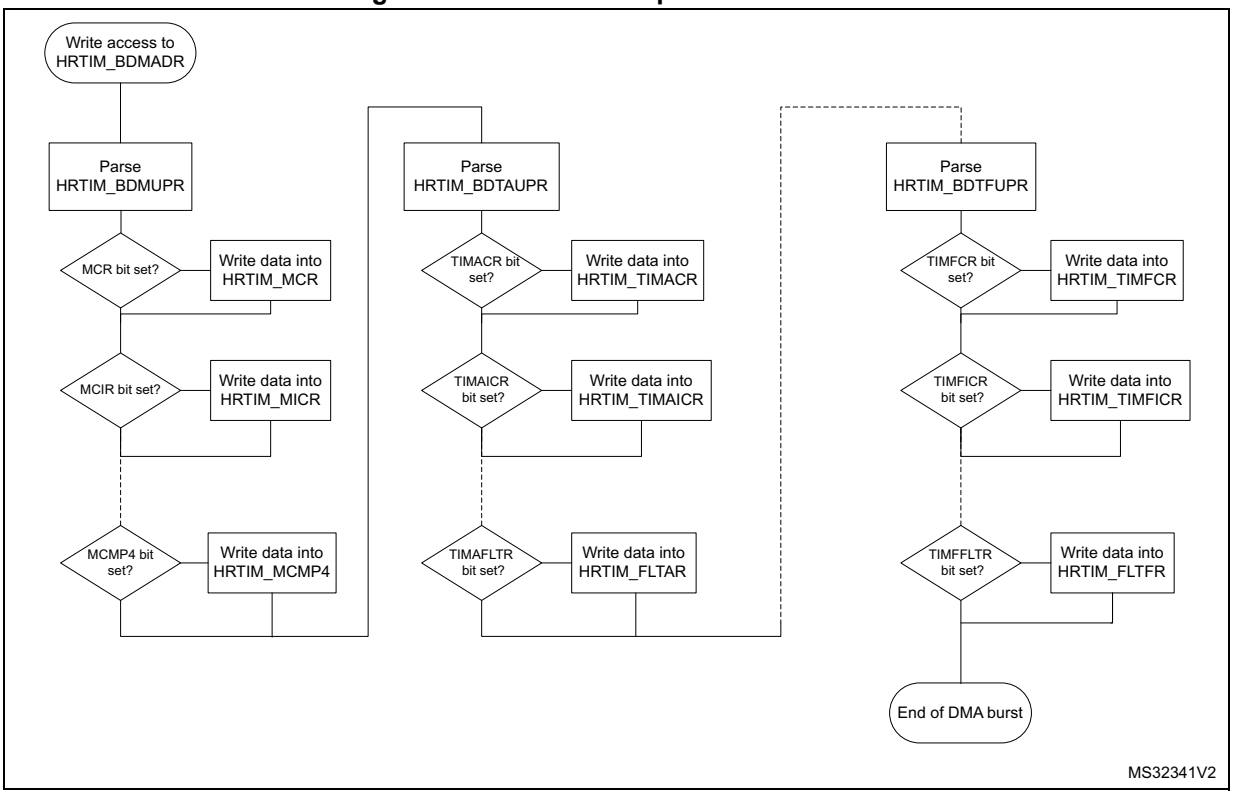
Note: Any trigger occurring while the burst is ongoing is discarded, except if it occurs during the very last data transfer.

The burst DMA mode is permanently enabled (there is no enabled bit). A burst DMA operation is started by the first write access into the HRTIM_BDMADR register.

It is only necessary to have the DMA controller pointing to the HRTIM_BDMADR register as the destination, in the memory, to the peripheral configuration with the peripheral increment mode disabled (the HRTIM handles internally the data rerouting to the final destination register).

To reinitialize the burst DMA mode if it was interrupted during a transaction, it is necessary to write at least to one of the six update registers.

Figure 446. Burst DMA operation flowchart



Several options are available once the DMA burst is completed, depending on the register update strategy.

If the PREEN bit is reset (preload disabled), the value written by the DMA is immediately transferred into the active register and the registers are updated sequentially, following the DMA transaction pace.

When the preload is enabled (PREEN bit set), there are three use cases:

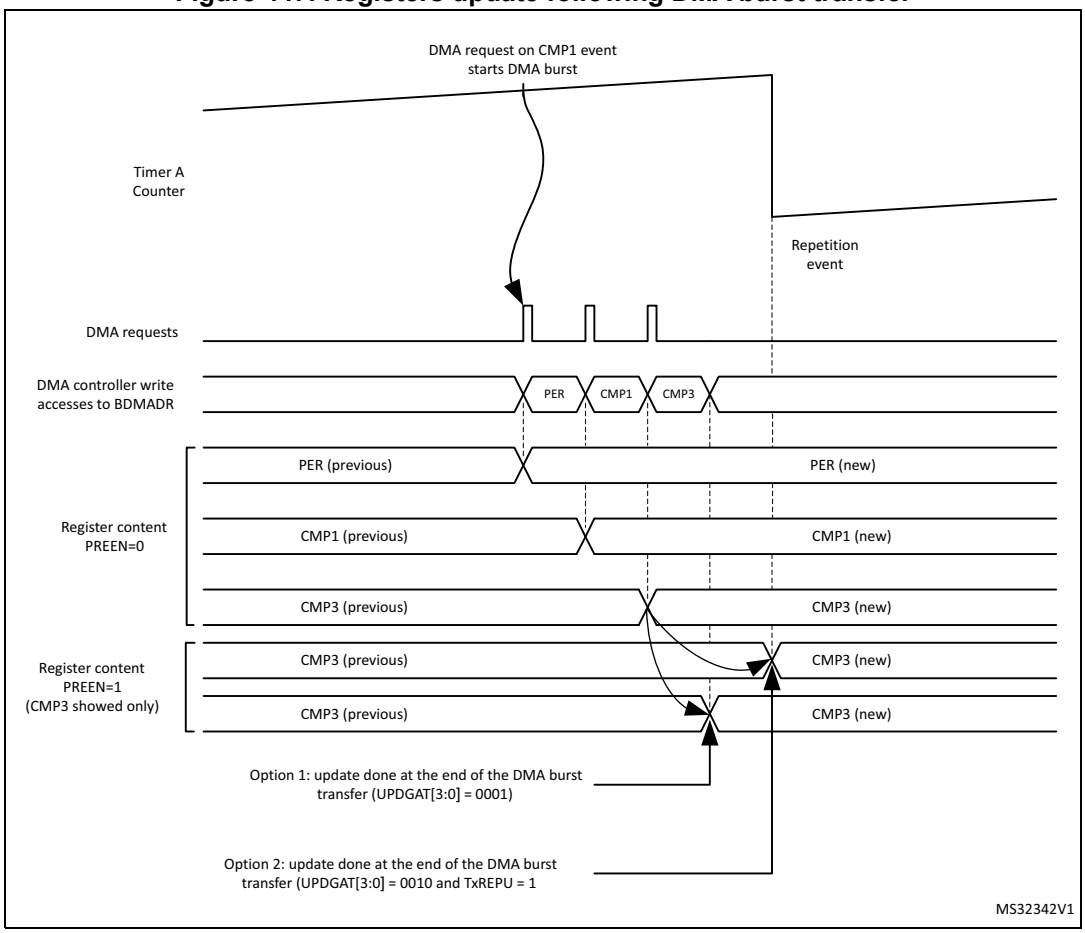
1. The update is done independently from DMA burst transfers (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 00 in HRTIM_MCR). In this case, and if it is necessary to have all transferred data taken into account simultaneously, the user must check that the DMA burst is completed before the update event takes place. On the contrary, if the update event happens while the DMA transfer is ongoing, only part of the registers is loaded and the complete register update requires two consecutive update events.
2. The update is done when the DMA burst transfer is completed (UPDGAT[3:0] = 0000 in HRTIM_TIMxCR and BRSTDMA[1:0] = 01 in HRTIM_MCR). This mode guarantees that all new register values are transferred simultaneously. This is done independently

from the counter value and can be combined with regular update events, if necessary (for instance, an update on a counter reset when TxRSTU is set).

3. The update is done on the update event following the DMA burst transfer completion (UPDGAT[3:0] = 0010 in HRTIM_TIMxCR and BRSTDMA[1:0] = 10 in HRTIM_MCR). This mode guarantees both a coherent update of all transferred data and the synchronization with regular update events, with the timer counter. In this case, if a regular update request occurs while the transfer is ongoing, it is discarded and the effective update happens on the next coming update request.

The chronogram on *Figure 447* presents the active register content for three cases: PREEN=0, UPDGAT[3:0] = 0001 and UPDGAT[3:0] = 0001 (when PREEN = 1).

Figure 447. Registers update following DMA burst transfer



40.3.24 HRTIM initialization

This section describes the recommended HRTIM initialization procedure, including other related MCU peripherals.

The HRTIM clock source must be enabled in the reset and clock control unit (RCC), while respecting the $_{fHRTIM}$ range for the DLL lock.

The DLL calibration must be started by setting CAL bit in HRTIM_DLLCR register.

The HRTIM master and timing units can be started only once the high-resolution unit is ready. This is indicated by the DLLRDY flag set. The DLLRDY flag can be polled before resuming the initialization or the calibration can run in background while other registers of the HRTIM or other MCU peripherals are initialized. In this case, the DLLRDY flag must be checked before starting the counters (an end-of-calibration interrupt can be issued if necessary, enabled with DLLRDYIE flag in HRTIM_IER). Once the DLL calibration is done, CALEN bit must be set to have it done periodically and compensate for potential voltage and temperature drifts. The calibration periodicity is defined using the CALRTE[1:0] bitfield in the HRTIM_DLLCR register.

The HRTIM control registers can be initialized as per the power converter topology and the timing units use case. All inputs have to be configured (source, polarity, edge-sensitivity).

The HRTIM outputs must be set up eventually, with the following sequence:

- The polarity must be defined using POLx bits in HRTIM_OUTxR
- The FAULT and IDLE states must be configured using FAULTx[1:0] and IDLESx bits in HRTIM_OUTxR

The HRTIM outputs are ready to be connected to the MCU I/Os. In the GPIO controller, the selected HRTIM I/Os have to be configured as per the alternate function mapping table in the product datasheet.

From this point on, the HRTIM controls the outputs, which are in the IDLE state.

The outputs are configured in RUN mode by setting TxyOEN bits in the HRTIM_OENR register. The two outputs are in the inactive state until the first valid set/reset event in RUN mode. Any output set/reset event (except software requests using SST, SRT) are ignored as long as TxCEN bit is reset, as well as burst mode requests (IDLEM bit value is ignored). Similarly, any counter reset request coming from the burst mode controller is ignored (if TxBM bit is set).

Note: When the deadtime insertion is enabled (DTEN bit set), it is necessary to force the output state by software, using SST and RST bits, to have the outputs in a complementary state as soon as the RUN mode is entered.

The HRTIM operation can eventually be started by setting TxCEN or MCEN bits in HRTIM_MCR.

If the HRTIM peripheral is reset with the reset and clock controller, the output control is released to the GPIO controller and the outputs are tri-stated.

40.3.25 Debug

When a microcontroller enters the debug mode (Cortex-M33 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_HRTIM_STOP configuration bit in DBG module:

- DBG_HRTIM_STOP = 0: no behavior change, the HRTIM continues to operate.
- DBG_HRTIM_STOP = 1: all HRTIM timers, including the master, are stopped. The outputs in RUN mode enter the FAULT state if FAULTx[1:0] = 01,10,11, or keep their current state if FAULTx[1:0] = 00. The outputs in idle state are maintained in this state. This is permanently maintained even if the MCU exits the halt mode. This is used to maintain a safe state during the execution stepping. The outputs can be enabled again by setting TxyOEN bit (requires the use of the debugger).

Timer behavior during MCU halt when DBG_HRTIM_STOP = 1

The set/reset crossbar, the dead-time and push-pull unit, the idle/balanced fault detection and all the logic driving the normal output in RUN mode are not affected by debug. The output keeps on toggling internally, so as to retrieve regular signals of the outputs when TxyOEN is set again (during or after the MCU halt). Associated triggers and filters are also following internal waveforms when the outputs are disabled.

FAULT inputs and events (any source) are enabled during the MCU halt.

Fault status bits can be set and TxyOEN bits reset during the MCU halt if a fault occurs at that time (TxyOEN and TxyODS are not affected by DBG_HRTIM_STOP bit state).

Synchronization, counter reset, start, and reset-start events are discarded in debug mode, as well as capture events. This is to keep all related registers stable as long as the MCU is halted.

The counter stops counting when a breakpoint is reached. However, the counter enable signal is not reset; consequently no start event is emitted when exiting from debug. All counter reset and capture triggers are disabled, as well as external events (ignored as long as the MCU is halted). The outputs SET and RST flags are frozen, except in case of forced software set/reset. A level-sensitive event is masked during the debug but is active again as soon as the debug is exited. For edge-sensitive events, if the signal is maintained active during the MCU halt, a new edge is not generated when exiting from debug.

The update events are discarded. This prevents any update trigger on hrtim_upd_en[3:1] inputs. DMA triggers are disabled. The burst mode circuit is frozen: the triggers are ignored and the burst mode counter stopped.

DLL calibration is not blocked while the MCU is halted (the DLLRDY flag can be set).

40.4 Application use cases

40.4.1 Buck converter

Buck converters are of common use as step-down converters. The HRTIM can control up to 12 buck converters with seven independent switching frequencies.

The converter usually operates at a fixed frequency and the Vin/Vout ratio depends on the duty cycle D applied to the power switch:

$$V_{out} = D \times V_{in}$$

The topology is given on *Figure 448* with the connection to the ADC for voltage reading.

Figure 448. Buck converter topology

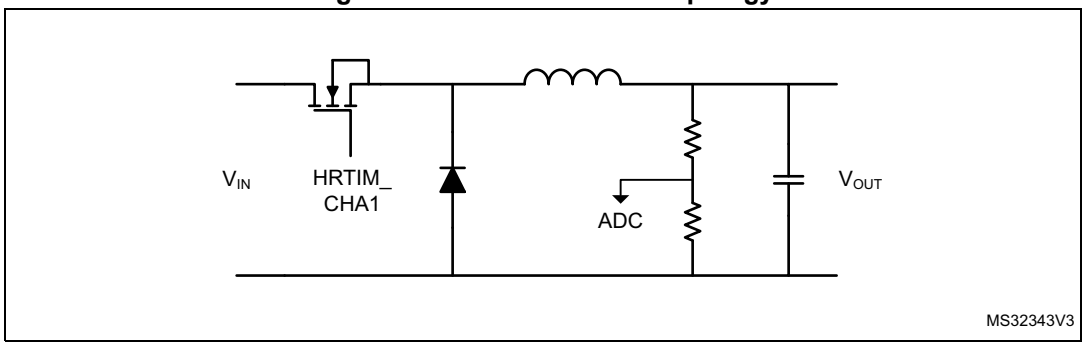
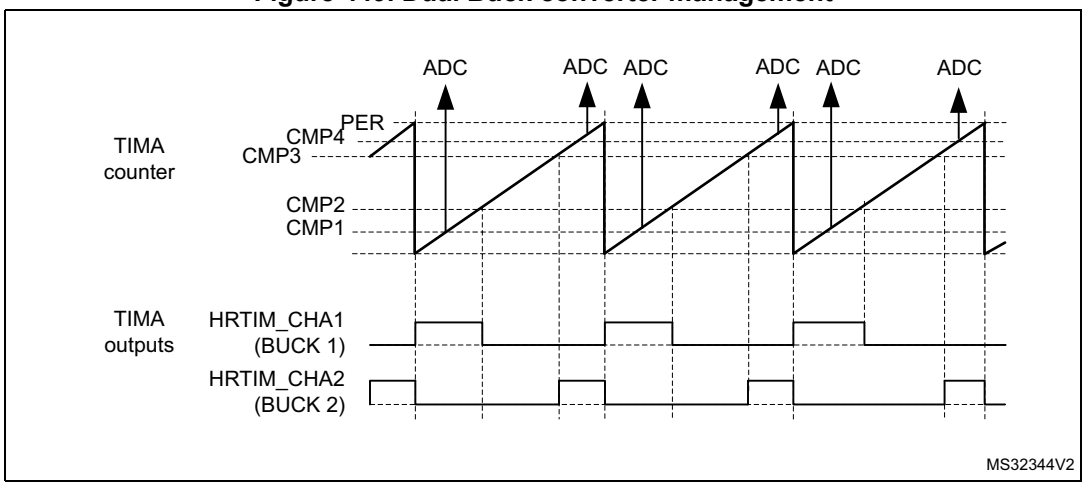


Figure 449 presents the management of two converters with identical frequency PWM signals. The outputs are defined as follows:

- HRTIM_CHA1 set on period, reset on CMP1
- HRTIM_CHA2 set on CMP3, reset on PER

The ADC is triggered twice per period, precisely in the middle of the ON time, using CMP2 and CMP4 events.

Figure 449. Dual Buck converter management

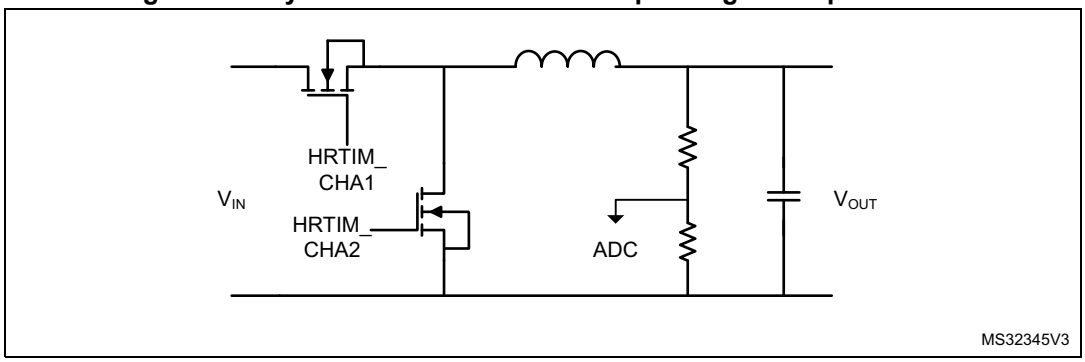


Timers A..E provide either 12 buck converters coupled by pairs (both with identical switching frequencies) or seven completely independent converters (each of them having a different switching frequency), using the master timer as the seventh time base.

40.4.2 Buck converter with synchronous rectification

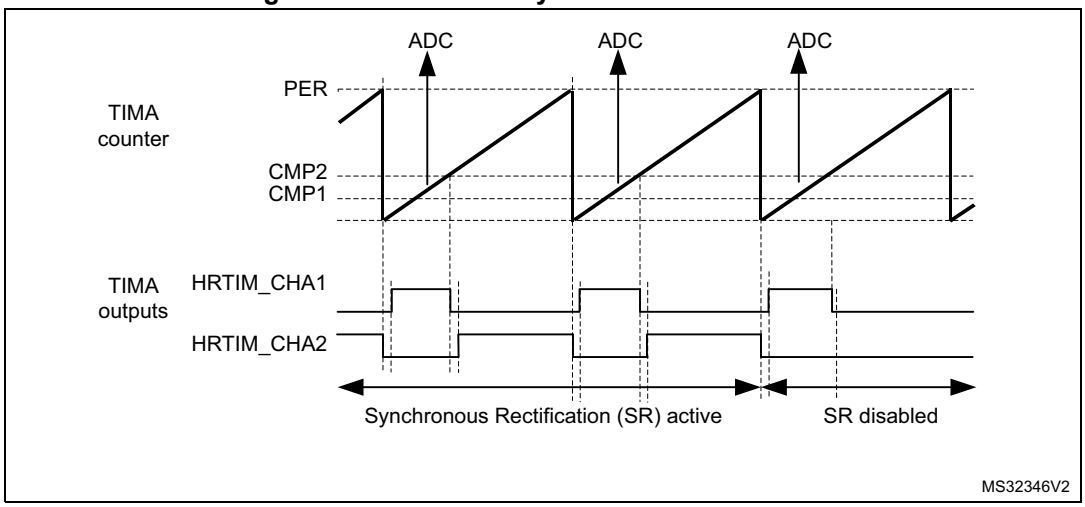
Synchronous rectification is used to minimize losses in buck converters, by means of a FET replacing the freewheeling diode. Synchronous rectification can be turned on or off on the fly depending on the output current level, as shown on *Figure 450*.

Figure 450. Synchronous rectification depending on output current



The main difference versus a single-switch buck converter is the addition of a deadtime for an almost complementary waveform generation on HRTIM_CHA2, based on the reference waveform on HRTIM_CHA1 (see *Figure 451*).

Figure 451. Buck with synchronous rectification



40.4.3 Multiphase converters

Multiphase techniques can be applied to multiple power conversion topologies (buck, flyback). Their main benefits are:

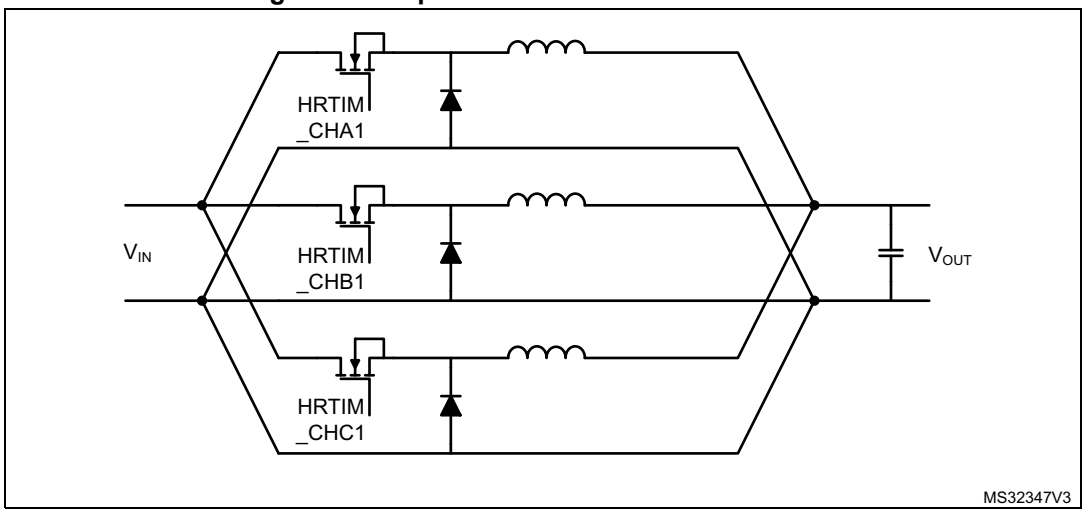
- Reduction of the current ripple on the input and output capacitors
- Reduced EMI
- Higher efficiency at light load by dynamically changing the number of phases (phase shedding)

The HRTIM manages multiple converters. The number of converters that can be controlled depends on the topologies and resources used (including the ADC triggers):

- Five buck converters with synchronous rectification (SR), using the master timer and the five timers
- Four buck converters (without SR), using the master timer and two timers

Figure 452 presents the topology of a 3-phase interleaved buck converter.

Figure 452. 3-phase interleaved buck converter



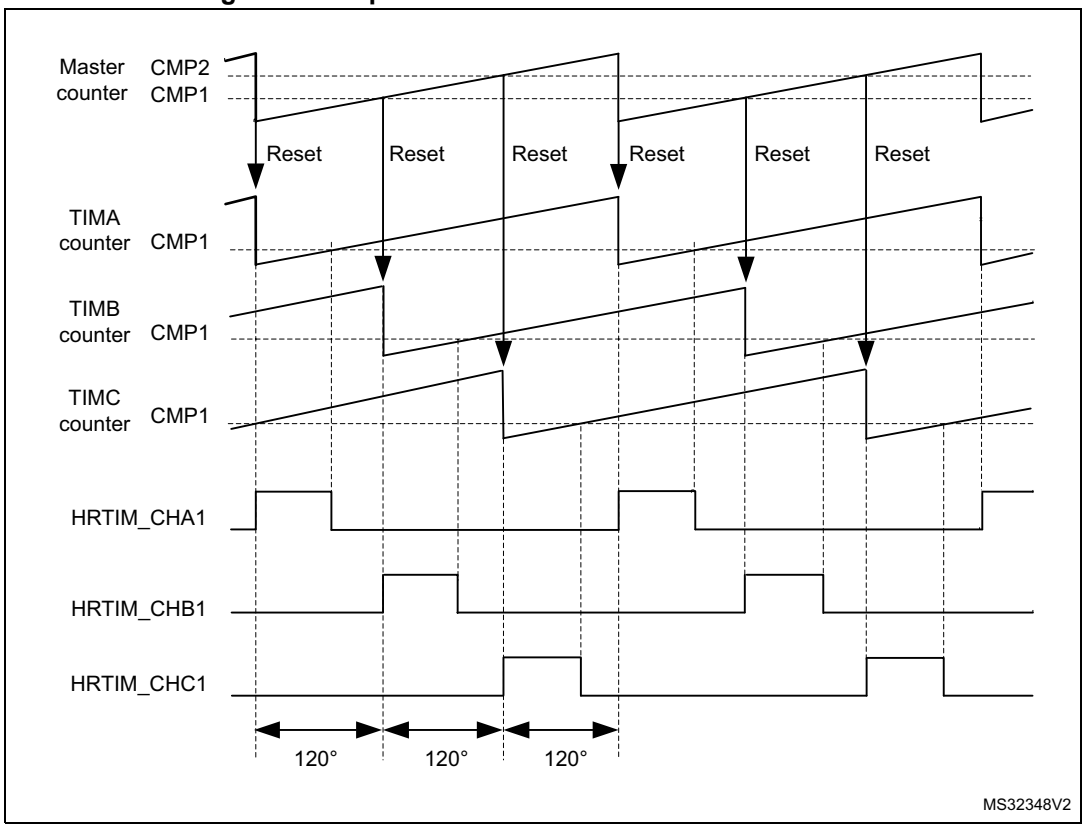
The master timer is responsible for the phase management: it defines the phase relationship between the converters by resetting the timers periodically. The phase-shift is 360° divided by the number of phases, 120° in the given example.

The duty cycle is then programmed into each of the timers. The outputs are defined as follows:

- HRTIM_CHA1 set on master timer period, reset on TACMP1
- HRTIM_CHB1 set on master timer MCMP1, reset on TBCMP1
- HRTIM_CHC1 set on master timer MCMP2, reset on TCCMP1

The ADC trigger can be generated on TxCMP2 compare event. Since all ADC trigger sources are phase-shifted because of the converter topology, it is possible to have all of them combined into a single ADC trigger to save ADC resources (for instance 1 ADC regular channel for the full multiphase converter).

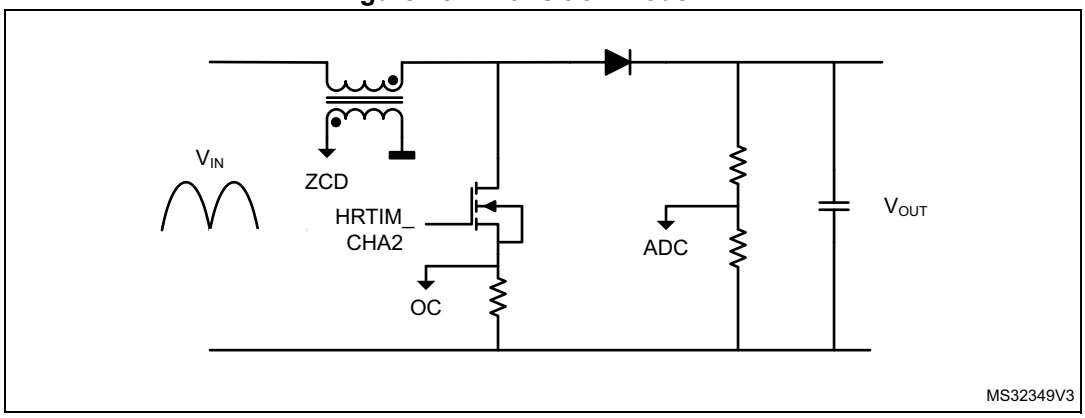
Figure 453. 3-phase interleaved buck converter control



40.4.4 Transition mode power factor correction

The basic operating principle is to build up current into an inductor during a fixed Ton time. This current then decays during the Toff time, and the period is restarted when it becomes null. This is detected using a zero crossing detection circuitry (ZCD), as shown on *Figure 454*. With a constant Ton time, the peak current value in the inductor is proportional to the rectified AC input voltage, which provides the power factor correction.

Figure 454. Transition mode PFC



This converter operates with a constant Ton time and a variable frequency due to the Toff time variation (depending on the input voltage). It must also include some features to operate when no zero-crossing is detected, or to limit the Ton time in case of overcurrent

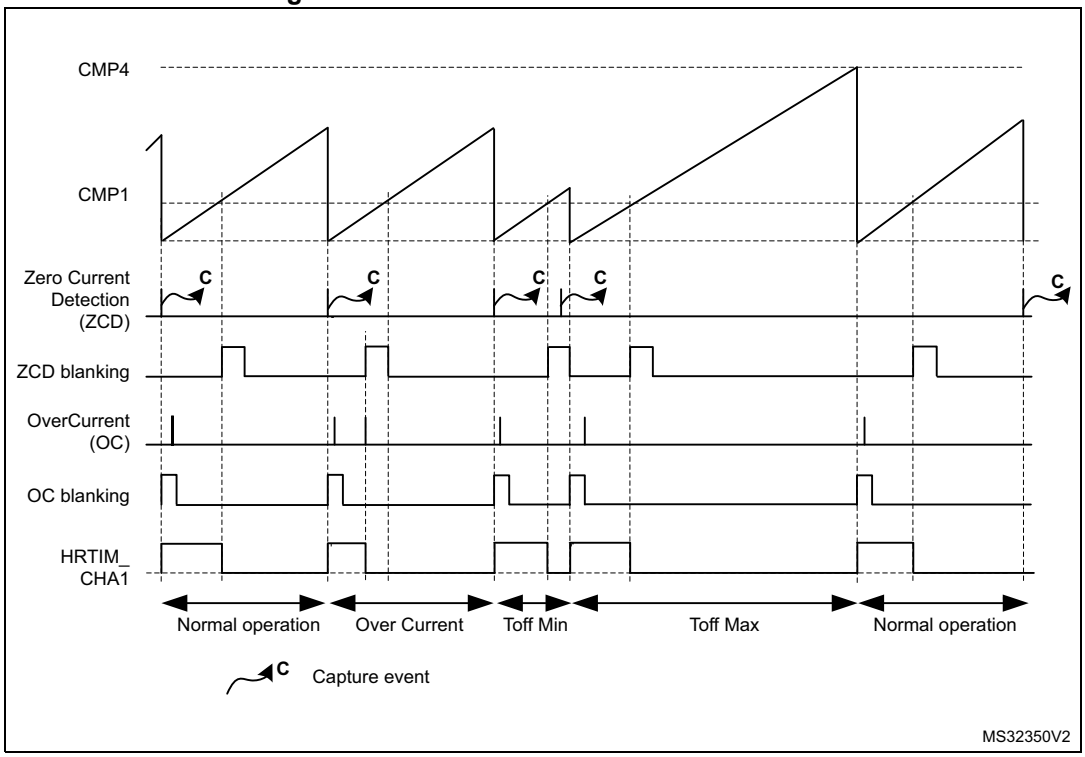
(OC). The OC feedback is usually conditioned with the built-in comparator and routed onto an external event input.

Figure 455 presents the waveform during the various operating modes, with the following defined parameters:

- Ton Min: masks spurious overcurrent (freewheeling diode recovery current), represented as OC blanking
- Ton Max: practically, the converter set-point. It is defined by CMP1
- Toff Min: limits the frequency when the current limit is close to zero (demagnetization is very fast). It is defined with CMP2.
- Toff Max: prevents the system to be stuck if no ZCD occurs. It is defined with CMP4 in autodelayed mode.

Both Toff values are autodelayed since the value must be relative to the output falling edge.

Figure 455. Transition mode PFC waveforms



40.5 HRTIM registers

40.5.1 HRTIM master timer control register (HRTIM_MCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	TFCEN	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN
rw	rw	rw		rw	rw	rw			rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		INTLVD[1:0]		HALF	RETRIG	CONT	CKPSC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **BRSTDMA[1:0]**: Burst DMA update

These bits define how the update occurs relatively to a burst DMA transaction.

00: Update done independently from the DMA burst transfer completion

01: Update done when the DMA burst transfer is completed

10: Update done on master timer roll-over following a DMA burst transfer completion. This mode only works in continuous mode.

11: Reserved

Bit 29 **MREPU**: Master timer repetition update

This bit defines whether an update occurs when the master timer repetition period is completed (either due to roll-over or reset events). MREPU can be set only if BRSTDMA[1:0] = 00 or 01.

0: Update on repetition disabled

1: Update on repetition enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **PREEN**: Preload enable

This bit enables the registers preload mechanism and defines whether the write accesses to the memory mapped registers are done into HRTIM's active or preload registers.

0: Preload disabled: the write access is directly done into the active register

1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]:** DAC synchronization

A DAC synchronization event can be enabled and generated when the master timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 40.3.21: DAC triggers* for connections details).

00: No DAC trigger generated

01: Trigger generated on hrtim_dac_trg1

10: Trigger generated on hrtim_dac_trg2

11: Trigger generated on hrtim_dac_trg3

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **TFCEN**: Timer F counter enable

This bit starts the timer F counter.

0: Timer F counter disabled

1: Timer F counter enabled

Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 21 **TECEN**: Timer E counter enable
This bit starts the timer E counter.
0: Timer E counter disabled
1: Timer E counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 20 **TDCEN**: Timer D counter enable
This bit starts the timer D counter.
0: Timer D counter disabled
1: Timer D counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 19 **TCCEN**: Timer C counter enable
This bit starts the timer C counter.
0: Timer C counter disabled
1: Timer C counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 18 **TBCEN**: Timer B counter enable
This bit starts the timer B counter.
0: Timer B counter disabled
1: Timer B counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 17 **TACEN**: Timer A counter enable
This bit starts the timer A counter.
0: Timer A counter disabled
1: Timer A counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bit 16 **MCEN**: Master timer counter enable
This bit starts the master timer counter.
0: Master counter disabled
1: Master counter enabled
Note: This bit must not be changed within a minimum of 8 cycles of f_{HRTIM} clock.

Bits 15:14 **SYNCSRC[1:0]**: Synchronization source
These bits define the source and event to be sent on the synchronization outputs SYNCOUT[2:1]
00: Master timer start
01: Master timer compare 1 event
10: Timer A start/reset
11: Timer A compare 1 event

Bits 13:12 **SYNCOUT[1:0]**: Synchronization output
These bits define the routing and conditioning of the synchronization output event.
00: Disabled
01: Reserved
10: Positive pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
11: Negative pulse on HRTIM_SCOUT output (16x f_{HRTIM} clock cycles)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 11 **SYNCSTRTM**: Synchronization starts master
This bit enables the master timer start when receiving a synchronization input event:
0: No effect on the master timer
1: A synchronization input event starts the master timer

Bit 10 **SYNCRSTM**: Synchronization resets master

This bit enables the master timer reset when receiving a synchronization input event:

0: No effect on the master timer

1: A synchronization input event resets the master timer

Bits 9:8 **SYNCIN[1:0]**: Synchronization input

These bits are defining the synchronization input source.

00: Disabled. HRTIM is not synchronized and runs in standalone mode.

01: Reserved

10: Internal event on hrtim_in_sync[2]: the HRTIM is synchronized with the on-chip timer (see *Section : Synchronization input*).

11: External event on hrtim_in_sync[3]: a positive pulse on HRTIM_SCIN input triggers the HRTIM.

Note: This parameter cannot be changed once the impacted timers are enabled.

Bits 7:6 **INTLVD[1:0]**: Interleaved mode

This bitfield is significant only when the HALF bit is reset. It enables the interleaved mode.

00: Interleaved mode disabled

01: Triple interleaved mode: when HRTIM_MPER register is written, the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/3 value, and the HRTIM_MCMP2R active register is automatically updated with 2x (HRTIM_MPER/3) value.

10: Quad interleaved mode: when HRTIM_MPER register is written, the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/4 value, the HRTIM_MCMP2R active register is automatically updated with HRTIM_MPER/2 value and the HRTIM_MCMP3R active register is automatically updated with 3x (HRTIM_MPER/4) value.

11: Interleaved mode disabled

Bit 5 **HALF**: Half mode

This bit enables the half duty-cycle mode: the HRTIM_MCMP1R active register is automatically updated with HRTIM_MPER/2 value when HRTIM_MPER register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: Re-triggerable mode

This bit defines the behavior of the master timer counter in single-shot mode.

0: The timer is not re-triggerable: a counter reset can be done only if the counter is stopped (period elapsed)

1: The timer is re-triggerable: a counter reset is done whatever the counter state (running or stopped)

Bit 3 **CONT**: Continuous mode

0: The timer operates in single-shot mode and stops when it reaches the MPER value

1: The timer operates in continuous (free-running) mode and rolls over to zero when it reaches the MPER value

Bits 2:0 **CKPSC[2:0]**: Clock prescaler

These bits define the master timer high-resolution clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{CKPSC[2:0]}$.

The prescaling ratio cannot be modified once the timer is enabled.

40.5.2 HRTIM master timer interrupt status register (HRTIM_MISR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPD	SYNC	MREP	MCMP 4	MCMP 3	MCMP 2	MCMP 1
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPD**: Master update interrupt flag
This bit is set by hardware when the master timer registers are updated.
0: No master update interrupt occurred
1: Master update interrupt occurred

Bit 5 **SYNC**: Sync input interrupt flag
This bit is set by hardware when a synchronization input event is received.
0: No sync input interrupt occurred
1: Sync input interrupt occurred

Bit 4 **MREP**: Master repetition interrupt flag
This bit is set by hardware when the master timer repetition period has elapsed.
0: No master repetition interrupt occurred
1: Master repetition interrupt occurred

Bit 3 **MCMP4**: Master compare 4 interrupt flag
Refer to MCMP1 description

Bit 2 **MCMP3**: Master compare 3 interrupt flag
Refer to MCMP1 description

Bit 1 **MCMP2**: Master compare 2 interrupt flag
Refer to MCMP1 description

Bit 0 **MCMP1**: Master compare 1 interrupt flag
This bit is set by hardware when the master timer counter matches the value programmed in the master compare 1 register.
0: No master compare 1 interrupt occurred
1: Master compare 1 interrupt occurred

40.5.3 HRTIM master timer interrupt clear register (HRTIM_MICR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **MUPDC**: Master update interrupt flag clear
Writing 1 to this bit clears the MUPDC flag in HRTIM_MISR register.

Bit 5 **SYNCC**: Sync input interrupt flag clear
Writing 1 to this bit clears the SYNC flag in HRTIM_MISR register.

Bit 4 **MREPC**: Repetition interrupt flag clear
Writing 1 to this bit clears the MREP flag in HRTIM_MISR register.

Bit 3 **MCMP4C**: Master compare 4 interrupt flag clear
Writing 1 to this bit clears the MCMP4 flag in HRTIM_MISR register.

Bit 2 **MCMP3C**: Master compare 3 interrupt flag clear
Writing 1 to this bit clears the MCMP3 flag in HRTIM_MISR register.

Bit 1 **MCMP2C**: Master compare 2 interrupt flag clear
Writing 1 to this bit clears the MCMP2 flag in HRTIM_MISR register.

Bit 0 **MCMP1C**: Master compare 1 interrupt flag clear
Writing 1 to this bit clears the MCMP1 flag in HRTIM_MISR register.

40.5.4 HRTIM master timer DMA interrupt enable register (HRTIM_MDIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **MUPDDE**: Master update DMA request enable
This bit is set and cleared by software to enable/disable the master update DMA requests.
0: Master update DMA request disabled
1: Master update DMA request enabled

Bit 21 **SYNCDE**: Sync input DMA request enable
This bit is set and cleared by software to enable/disable the sync input DMA requests.
0: Sync input DMA request disabled
1: Sync input DMA request enabled

Bit 20 **MREPDE**: Master repetition DMA request enable
This bit is set and cleared by software to enable/disable the master timer repetition DMA requests.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **MCMP4DE**: Master compare 4 DMA request enable
Refer to MCMP1DE description

Bit 18 **MCMP3DE**: Master compare 3 DMA request enable
Refer to MCMP1DE description

Bit 17 **MCMP2DE**: Master compare 2 DMA request enable
Refer to MCMP1DE description

Bit 16 **MCMP1DE**: Master compare 1 DMA request enable
This bit is set and cleared by software to enable/disable the master timer compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **MUPDIE**: Master update interrupt enable
This bit is set and cleared by software to enable/disable the master timer registers update interrupts
0: Master update interrupts disabled
1: Master update interrupts enabled

Bit 5 **SYNCIE**: Sync input interrupt enable
This bit is set and cleared by software to enable/disable the sync input interrupts
0: Sync input interrupts disabled
1: Sync input interrupts enabled

Bit 4 **MREPIE**: Master repetition interrupt enable
This bit is set and cleared by software to enable/disable the master timer repetition interrupts
0: Master repetition interrupt disabled
1: Master repetition interrupt enabled

Bit 3 **MCMP4IE**: Master compare 4 interrupt enable
Refer to MCMP1IE description

Bit 2 **MCMP3IE**: Master compare 3 interrupt enable
Refer to MCMP1IE description

Bit 1 **MCMP2IE**: Master compare 2 interrupt enable
Refer to MCMP1IE description

Bit 0 **MCMP1IE**: Master compare 1 interrupt enable
This bit is set and cleared by software to enable/disable the master timer compare 1 interrupt
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

40.5.5 HRTIM master timer counter register (HRTIM_MCNTR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCNT[15:0]**: Counter value

Holds the master timer counter value. This register can only be written when the master timer is stopped (MCEN = 0 in HRTIM_MCR).

Note: For HR clock prescaling ratio below 32 (CKPSCCKPSC[2:0] < 5), the least significant bits of the counter are not significant.

Note: The timer behavior is not guaranteed if the counter value is set above the HRTIM_MPER register value.

40.5.6 HRTIM master timer period register (HRTIM_MPER)

Address offset: 0x014

Reset value: 0x0000 FFDF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MPER[15:0]**: Master timer period value

This register defines the counter overflow value.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The maximum value is 0x0000 FFDF.

40.5.7 HRTIM master timer repetition register (HRTIM_MREP)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MREP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **MREP[7:0]**: Master timer repetition period value

This register holds the repetition period value for the master counter. It is either the preload register or the active register if preload is disabled.

40.5.8 HRTIM master timer compare 1 register (HRTIM_MCMP1R)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP1[15:0]**: Master timer compare 1 value

This register holds the master timer compare 1 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

40.5.9 HRTIM master timer compare 2 register (HRTIM_MCMP2R)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP2[15:0]**: Master timer compare 2 value

This register holds the master timer compare 2 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

40.5.10 HRTIM master timer compare 3 register (HRTIM_MCMP3R)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP3[15:0]**: Master timer compare 3 value

This register holds the master timer compare 3 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

40.5.11 HRTIM master timer compare 4 register (HRTIM_MCMP4R)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMP4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MCMP4[15:0]**: Master timer compare 4 value

This register holds the master timer compare 4 value. It is either the preload register or the active register if preload is disabled.

The compare value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

40.5.12 HRTIM timer x control register (HRTIM_TIMxCR) (x = A to F)

Address offset: Block A: 0x100

Address offset: Block B: 0x200

Address offset: Block C: 0x300

Address offset: Block D: 0x400

Address offset: Block E: 0x500

Address offset: Block F: 0x600

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDGAT[3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	TAU	TxRSTU	TxREPU	TFU
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	RSYNCU	INTLVD[1:0]		PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **UPDGAT[3:0]**: Update gating

These bits define how the update occurs relatively to the burst DMA transaction and the external update request on update enable inputs hrtim_upd_en[3:1] (see *Table 412*).

The update events, as mentioned below, can be: MSTU, TFU, TEU, TDU, TCU, TBU, TAU, TxRSTU, TxREPU.

0000: The update occurs independently from the DMA burst transfer

0001: The update occurs when the DMA burst transfer is completed

0010: The update occurs on the update event following the DMA burst transfer completion

0011: The update occurs on a rising edge on hrtim_upd_en1

0100: The update occurs on a rising edge on hrtim_upd_en2

0101: The update occurs on a rising edge on hrtim_upd_en3

0110: The update occurs on the update event following a rising edge on hrtim_upd_en1

0111: The update occurs on the update event following a rising edge on hrtim_upd_en2

1000: The update occurs on the update event following a rising edge on hrtim_upd_en3

Others: Reserved

Note: This bitfield must be reset before programming a new value.

For UPDGAT[3:0] values equal to 0001, 0011, 0100, 0101, it is possible to have multiple concurrent update source (for instance RSTU and DMA burst).

Bit 27 **PREEN**: Preload enable

This bit enables the registers preload mechanism and defines whether a write access into a preload-able register is done into the active or the preload register.

0: Preload disabled: the write access is directly done into the active register

1: Preload enabled: the write access is done into the preload register

Bits 26:25 **DACSYNC[1:0]**: DAC synchronization

A DAC synchronization event is generated when the timer update occurs. These bits are defining on which output the DAC synchronization is sent (refer to *Section 40.3.21: DAC triggers* for connections details).

00: No DAC trigger generated
01: Trigger generated on hrtim_dac_trg1
10: Trigger generated on hrtim_dac_trg2
11: Trigger generated on hrtim_dac_trg3

Bit 24 **MSTU**: Master timer update

Register update is triggered by the master timer update.
0: Update by master timer disabled
1: Update by master timer enabled

Bit 23 **TEU**: Timer E update

Register update is triggered by the timer E update
0: Update by timer E disabled
1: Update by timer E enabled

Note: This bit is reserved for HRTIM_TIMECR. *It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMFCR.*

Bit 22 **TDU**: Timer D update

Register update is triggered by the timer D update
0: Update by timer D disabled
1: Update by timer D enabled

Note: This bit is reserved for HRTIM_TIMDCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 21 **TCU**: Timer C update

Register update is triggered by the timer C update
0: Update by timer C disabled
1: Update by timer C enabled

Note: This bit is reserved for HRTIM_TIMCCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 20 **TBU**: Timer B update

Register update is triggered by the timer B update
0: Update by timer B disabled
1: Update by timer B enabled

Note: This bit is reserved for HRTIM_TIMBCR. It is only available for HRTIM_TIMACR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 19 **TAU**: Timer A update

Register update is triggered by the timer A update
0: Update by timer A disabled
1: Update by timer A enabled

Note: This bit is reserved for HRTIM_TIMBCR. It is only available for HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR, HRTIM_TIMFCR.

Bit 18 **TxRSTU**: Timer x reset update

Register update is triggered by timer x counter reset or roll-over to 0 after reaching the period value in continuous mode.
0: Update by timer x reset / roll-over disabled
1: Update by timer x reset / roll-over enabled

Bit 17 **TxREPU**: Timer x repetition update

Register update is triggered when the counter rolls over and HRTIM_REPx = 0

0: Update on repetition disabled

1: Update on repetition enabled

Bit 16 **TFU**: Timer F update

Register update is triggered by the timer F update

0: Update by timer F disabled

1: Update by timer F enabled

Note: This bit is reserved for HRTIM_TIMFCR. It is only available for HRTIM_TIMACR, HRTIM_TIMBCR, HRTIM_TIMCCR, HRTIM_TIMDCR, HRTIM_TIMECR.

Bits 15:14 **DELCMP4[1:0]**: CMP4 auto-delayed mode

This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Section : Autodelayed mode*).

00: CMP4 register is always active (standard compare mode)

01: CMP4 value is recomputed and is active following a capture 2 event

10: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 1 match (timeout function if capture 2 event is missing)

11: CMP4 value is recomputed and is active following a capture 2 event, or is recomputed and active after Compare 3 match (timeout function if capture event is missing)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bits 13:12 **DELCMP2[1:0]**: CMP2 auto-delayed mode

This bitfield defines whether the compare register is behaving in standard mode (compare match issued as soon as counter equal compare), or in auto-delayed mode (see *Section : Autodelayed mode*).

00: CMP2 register is always active (standard compare mode)

01: CMP2 value is recomputed and is active following a capture 1 event

10: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after compare 1 match (timeout function if capture event is missing)

11: CMP2 value is recomputed and is active following a capture 1 event, or is recomputed and active after compare 3 match (timeout function if capture event is missing)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 11 **SYNCSTRTx**: Synchronization starts timer x

This bit defines the timer x behavior following the synchronization event:

0: No effect on timer x

1: A synchronization input event starts the timer x

Bit 10 **SYNCRSTx**: Synchronization resets timer x

This bit defines the timer x behavior following the synchronization event:

0: No effect on timer x

1: A synchronization input event resets the timer x

Bit 9 **RSYNCU**: Re-synchronized update

This bit specifies whether update source coming outside from the timing unit must be synchronized:

0: The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) is taken into account immediately

1: The update coming from adjacent timers (when MSTU, TAU, TBU, TCU, TDU, TEU, TFU bit is set) or from a software update (TxSWU bit) is taken into account on the following reset/roll-over event.

Note: This bit is significant only when UPDGAT[3:0] = 0000, it is ignored otherwise.

Bits 8:7 **INTLVD[1:0]**: Interleaved mode

This bitfield is significant only when the HALF bit is reset. It enables the interleaved mode.

00: Interleaved mode disabled

01: Triple interleaved mode: when HRTIM_PERxR register is written, the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/3 value, and the HRTIM_CMP2xR active register is automatically updated with 2x (HRTIM_PERxR/3) value.

10: Quad interleaved mode: when HRTIM_PERxR register is written, the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/4 value, the HRTIM_CMP2xR active register is automatically updated with HRTIM_PERxR/2 value and the HRTIM_CMP3xR active register is automatically updated with 3x (HRTIM_PERxR/4) value.

11: Interleaved mode disabled

Bit 6 **PSHPLL**: Push-pull mode enable

This bit enables the push-pull mode.

0: Push-pull mode disabled

1: Push-pull mode enabled

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 5 **HALF**: Half mode enable

This bit enables the half duty-cycle mode: the HRTIM_CMP1xR active register is automatically updated with HRTIM_PERxR/2 value when HRTIM_PERxR register is written.

0: Half mode disabled

1: Half mode enabled

Bit 4 **RETRIG**: Re-triggerable mode

This bit defines the counter behavior in single shot mode.

0: The timer is not re-triggerable: a counter reset is done if the counter is stopped (period elapsed in single-shot mode or counter stopped in continuous mode)

1: The timer is re-triggerable: a counter reset is done whatever the counter state.

Bit 3 **CONT**: Continuous mode

This bit defines the timer operating mode.

0: The timer operates in single-shot mode and stops when it reaches TIMxPER value

1: The timer operates in continuous mode and rolls over to zero when it reaches TIMxPER value

Bits 2:0 **CKPSCx[2:0]**: HRTIM timer x clock prescaler

These bits define the master timer high-resolution clock prescaler ratio.

The counter clock equivalent frequency ($f_{COUNTER}$) is equal to $f_{HRCK} / 2^{CKPSC[2:0]}$.

The prescaling ratio cannot be modified once the timer is enabled.

40.5.13 HRTIM timer x interrupt status register (HRTIM_TIMxISR) (x = A to F)

Address offset: Block A: 0x104

Address offset: Block B: 0x204

Address offset: Block C: 0x304

Address offset: Block D: 0x404

Address offset: Block E: 0x504

Address offset: Block F: 0x604

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COTS	COS	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORDE	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	COT	REP	CMP4	CMP3	CMP2	CMP1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **COTS**: Crossbar Override Trigger status

This status bit is a raw copy of the crossbar override trigger input. It can be polled to verify if the crossbar override mode can be rearmed

0: Crossbar Override trigger is inactive

1: Crossbar Override trigger is active

Bit 22 **COS**: Crossbar Override status

This status bit indicates the crossbar override status.

0: Crossbar Override is inactive (crossbar is active)

1: Crossbar Override is active

Bit 21 **O2CPY**: Output 2 copy

This status bit is a raw copy of the output 2 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 2 is inactive

1: Output 2 is active

Bit 20 **O1CPY**: Output 1 copy

This status bit is a raw copy of the output 1 state, before the output stage (chopper, polarity). It allows to check the current output state before re-enabling the output after a delayed protection.

0: Output 1 is inactive

1: Output 1 is active

Bit 19 **O2STAT**: Output 2 status

This status bit indicates the output 2 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 2 was inactive

1: Output 2 was active

Bit 18 **O1STAT**: Output 1 status

This status bit indicates the output 1 state when the delayed idle protection was triggered. This bit is updated upon any new delayed protection entry. This bit is not updated in balanced idle.

0: Output 1 was inactive

1: Output 1 was active

Bit 17 **IPPSTAT**: Idle push-pull Status

This status bit indicates on which output the signal was applied, in push-pull mode balanced fault mode or delayed idle mode, when the protection was triggered (whatever the output state, active or inactive).

0: Protection occurred when the output 1 was active and output 2 forced inactive

1: Protection occurred when the output 2 was active and output 1 forced inactive

Bit 16 **CPPSTAT**: Current push-pull status

This status bit indicates on which output the signal is currently applied, in push-pull mode. It is only significant in this configuration.

0: Signal applied on output 1 and output 2 forced inactive

1: Signal applied on output 2 and output 1 forced inactive

Bit 15 **CORDE**: Crossbar Override Rearm Delay Elapsed flag

This bit is set when the crossbar override mode can be rearmed. This interrupt flag is asserted when at least on of the one of the 2 channels is configured to Delayed fault force modes (COFMx[2:0] = 000 and 001).

0: No Crossbar override rearm delay elapsed interrupt occurred

1: Crossbar override rearm delay elapsed interrupt occurred

Bit 14 **DLYPRT**: Delayed protection flag

0: No delayed protection interrupt occured

1: Delayed Idle or balanced Idle mode entry occured

Bit 13 **RST**: Reset and/or roll-over interrupt flag

This bit is set by hardware when the timer x counter is reset or rolls over in continuous mode.

0: No TIMx counter reset/roll-over interrupt occurred

1: TIMX counter reset/roll-over interrupt occurred

Bit 12 **RSTx2**: Output 2 reset interrupt flag timer x

Refer to RSTx1 description

Bit 11 **SETx2**: Output 2 set interrupt flag timer x

Refer to SETx1 description

Bit 10 **RSTx1**: Output 1 reset interrupt flag timer x

This bit is set by hardware when the Tx1 output is reset (goes from active to inactive mode).

0: No Tx1 output reset interrupt occurred

1: Tx1 output reset interrupt occurred

Bit 9 **SETx1**: Output 1 set interrupt flag timer x

This bit is set by hardware when the Tx1 output is set (goes from inactive to active mode).

0: No Tx1 output set interrupt occurred

1: Tx1 output set interrupt occurred

Bit 8 **CPT2**: Capture 2 interrupt flag

Refer to CPT1 description

Bit 7 **CPT1**: Capture 1 interrupt flag

This bit is set by hardware when the timer x capture 1 event occurs.

0: No timer x capture 1 interrupt occurred

1: Timer x capture 1 interrupt occurred

Bit 6 **UPD**: Update interrupt flag
This bit is set by hardware when the timer x update event occurs.
0: No timer x update interrupt occurred
1: Timer x update interrupt occurred

Bit 5 **COT**: Crossbar Override Trigger flag
This bit is set when the crossbar override mode is triggered.
0: No Crossbar override trigger interrupt occurred
1: Crossbar override trigger interrupt occured

Bit 4 **REP**: Repetition interrupt flag
This bit is set by hardware when the timer x repetition period has elapsed.
0: No timer x repetition interrupt occurred
1: Timer x repetition interrupt occurred

Bit 3 **CMP4**: Compare 4 interrupt flag
Refer to CMP1 description

Bit 2 **CMP3**: Compare 3 interrupt flag
Refer to CMP1 description

Bit 1 **CMP2**: Compare 2 interrupt flag
Refer to CMP1 description

Bit 0 **CMP1**: Compare 1 interrupt flag
This bit is set by hardware when the timer x counter matches the value programmed in the compare 1 register.
0: No compare 1 interrupt occurred
1: Compare 1 interrupt occurred

40.5.14 HRTIM timer x interrupt clear register (HRTIM_TIMxICR) (x = A to F)

Address offset: Block A: 0x108

Address offset: Block B: 0x208

Address offset: Block C: 0x308

Address offset: Block D: 0x408

Address offset: Block E: 0x508

Address offset: Block F: 0x608

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORDEC	DLYPRTC	RSTC	RSTx2C	SETx2C	RSTx1C	SETx1C	CPT2C	CPT1C	UPDC	COTC	REPC	CMP4C	CMP3C	CMP2C	CMP1C
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CORDEC**: Crossbar Override Rearm Delay Elapsed flag clear
Writing 1 to this bit clears the CORDE flag in HRTIM_TIMxISR register

Bit 14 **DLYPRTC**: Delayed protection flag clear
Writing 1 to this bit clears the DLYPRT flag in HRTIM_TIMxISR register

Bit 13 **RSTC**: Reset interrupt flag clear
Writing 1 to this bit clears the RST flag in HRTIM_TIMxISR register

Bit 12 **RSTx2C**: Output 2 reset flag clear timer x
Writing 1 to this bit clears the RSTx2 flag in HRTIM_TIMxISR register

Bit 11 **SETx2C**: Output 2 set flag clear timer x
Writing 1 to this bit clears the SETx2 flag in HRTIM_TIMxISR register

Bit 10 **RSTx1C**: Output 1 reset flag clear timer x
Writing 1 to this bit clears the RSTx1 flag in HRTIM_TIMxISR register

Bit 9 **SETx1C**: Output 1 set flag clear timer x
Writing 1 to this bit clears the SETx1 flag in HRTIM_TIMxISR register

Bit 8 **CPT2C**: Capture 2 interrupt flag clear
Writing 1 to this bit clears the CPT2 flag in HRTIM_TIMxISR register

Bit 7 **CPT1C**: Capture 1 interrupt flag clear
Writing 1 to this bit clears the CPT1 flag in HRTIM_TIMxISR register

Bit 6 **UPDC**: Update interrupt flag clear
Writing 1 to this bit clears the UPD flag in HRTIM_TIMxISR register

Bit 5 **COTC**: Crossbar Override Trigger flag clear
Writing 1 to this bit clears the COT flag in HRTIM_TIMxISR register

Bit 4 **REPC**: Repetition interrupt flag clear
Writing 1 to this bit clears the REP flag in HRTIM_TIMxISR register

Bit 3 **CMP4C**: Compare 4 interrupt flag clear
Writing 1 to this bit clears the CMP4 flag in HRTIM_TIMxISR register

Bit 2 **CMP3C**: Compare 3 interrupt flag clear
Writing 1 to this bit clears the CMP3 flag in HRTIM_TIMxISR register

Bit 1 **CMP2C**: Compare 2 interrupt flag clear
Writing 1 to this bit clears the CMP2 flag in HRTIM_TIMxISR register

Bit 0 **CMP1C**: Compare 1 interrupt flag clear
Writing 1 to this bit clears the CMP1 flag in HRTIM_TIMxISR register

40.5.15 HRTIM timer x DMA interrupt enable register (HRTIM_TIMxDIER) (x = A to F)

Address offset: Block A: 0x10C

Address offset: Block B: 0x20C

Address offset: Block C: 0x30C

Address offset: Block D: 0x40C

Address offset: Block E: 0x50C

Address offset: Block F: 0x60C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CORDEDE	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SETx1DE	CPT2DE	CPT1DE	UPDDE	COTDE	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORDEIE	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SETx1IE	CPT2IE	CPT1IE	UPDIE	COTIE	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CORDEDE**: Crossbar override rearm delay elapsed DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on Crossbar override rearm delay elapsed event.
0: Crossbar override rearm delay elapsed DMA request disabled
1: Crossbar override rearm delay elapsed DMA request enabled

Bit 30 **DLYPRTDE**: Delayed protection DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on delayed protection.
0: Delayed protection DMA request disabled
1: Delayed protection DMA request enabled

Bit 29 **RSTDE**: Reset/roll-over DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over DMA request disabled
1: Timer x counter reset/roll-over DMA request enabled

Bit 28 **RSTx2DE**: Output 2 reset DMA request enable timer x
Refer to RSTx1DE description

Bit 27 **SETx2DE**: Output 2 set DMA request enable timer x
Refer to SETx1DE description

Bit 26 **RSTx1DE**: Output 1 reset DMA request enable timer x
This bit is set and cleared by software to enable/disable Tx1 output reset DMA requests.
0: Tx1 output reset DMA request disabled
1: Tx1 output reset DMA request enabled

Bit 25 **SETx1DE**: Output 1 set DMA request enable timer x
This bit is set and cleared by software to enable/disable Tx1 output set DMA requests.
0: Tx1 output set DMA request disabled
1: Tx1 output set DMA request enabled

Bit 24 **CPT2DE**: Capture 2 DMA request enable
Refer to CPT1DE description

Bit 23 **CPT1DE**: Capture 1 DMA request enable
This bit is set and cleared by software to enable/disable capture 1 DMA requests.
0: Capture 1 DMA request disabled
1: Capture 1 DMA request enabled

Bit 22 **UPDDE**: Update DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on update event.
0: Update DMA request disabled
1: Update DMA request enabled

Bit 21 **COTDE**: Crossbar Override Trigger DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on Crossbar Override Trigger event.
0: Crossbar Override Trigger DMA request disabled
1: Crossbar Override Trigger DMA request enabled

Bit 20 **REPDE**: Repetition DMA request enable
This bit is set and cleared by software to enable/disable DMA requests on repetition event.
0: Repetition DMA request disabled
1: Repetition DMA request enabled

Bit 19 **CMP4DE**: Compare 4 DMA request enable
Refer to CMP1DE description

Bit 18 **CMP3DE**: Compare 3 DMA request enable
Refer to CMP1DE description

Bit 17 **CMP2DE**: Compare 2 DMA request enable
Refer to CMP1DE description

Bit 16 **CMP1DE**: Compare 1 DMA request enable
This bit is set and cleared by software to enable/disable the compare 1 DMA requests.
0: Compare 1 DMA request disabled
1: Compare 1 DMA request enabled

Bit 15 **CORDEIE**: Crossbar override rearm delay elapsed interrupt enable
This bit is set and cleared by software to enable/disable interrupts on Crossbar override rearm delay elapsed event.
0: Crossbar override rearm delay elapsed interrupts disabled
1: Crossbar override rearm delay elapsed interrupts enabled

Bit 14 **DLYPRTIE**: Delayed protection interrupt enable
This bit is set and cleared by software to enable/disable interrupts on delayed protection.
0: Delayed protection interrupts disabled
1: Delayed protection interrupts enabled

Bit 13 **RSTIE**: Reset/roll-over interrupt enable
This bit is set and cleared by software to enable/disable interrupts on timer x counter reset or roll-over in continuous mode.
0: Timer x counter reset/roll-over interrupt disabled
1: Timer x counter reset/roll-over interrupt enabled

Bit 12 **RSTx2IE**: Output 2 reset interrupt enable timer x
Refer to RSTx1IE description

Bit 11 **SETx2IE**: Output 2 set interrupt enable timer x
Refer to SETx1IE description

Bit 10 **RSTx1IE**: Output 1 reset interrupt enable timer x
This bit is set and cleared by software to enable/disable Tx1 output reset interrupts.
0: Tx1 output reset interrupts disabled
1: Tx1 output reset interrupts enabled

Bit 9 **SETx1IE**: Output 1 set interrupt enable timer x
This bit is set and cleared by software to enable/disable Tx1 output set interrupts.
0: Tx1 output set interrupts disabled
1: Tx1 output set interrupts enabled

Bit 8 **CPT2IE**: Capture interrupt enable
Refer to CPT1IE description

Bit 7 **CPT1IE**: Capture interrupt enable
This bit is set and cleared by software to enable/disable capture 1 interrupts.
0: Capture 1 interrupts disabled
1: Capture 1 interrupts enabled

Bit 6 **UPDIE**: Update interrupt enable
This bit is set and cleared by software to enable/disable update event interrupts.
0: Update interrupts disabled
1: Update interrupts enabled

Bit 5 **COTIE**: Crossbar Override Trigger interrupt enable
This bit is set and cleared by software to enable/disable Crossbar Override Trigger interrupts.
0: Crossbar Override Trigger interrupts disabled
1: Crossbar Override Trigger enabled

Bit 4 **REPIE**: Repetition interrupt enable
This bit is set and cleared by software to enable/disable repetition event interrupts.
0: Repetition interrupts disabled
1: Repetition interrupts enabled

Bit 3 **CMP4IE**: Compare 4 interrupt enable
Refer to CMP1IE description

Bit 2 **CMP3IE**: Compare 3 interrupt enable
Refer to CMP1IE description

Bit 1 **CMP2IE**: Compare 2 interrupt enable
Refer to CMP1IE description

Bit 0 **CMP1IE**: Compare 1 interrupt enable
This bit is set and cleared by software to enable/disable the compare 1 interrupts.
0: Compare 1 interrupt disabled
1: Compare 1 interrupt enabled

40.5.16 HRTIM timer x counter register (HRTIM_CNTxR) (x = A to F)

Address offset: Block A: 0x110

Address offset: Block B: 0x210

Address offset: Block C: 0x310

Address offset: Block D: 0x410

Address offset: Block E: 0x510

Address offset: Block F: 0x610

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNTx[15:0]**: Timer x counter value

This register holds the timer x counter value. It can only be written when the timer is stopped (TxCEN = 0 in HRTIM_TIMxCR).

Note: For HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

Note: The timer behavior is not guaranteed if the counter value is above the HRTIM_PERxR register value.

40.5.17 HRTIM timer x period register (HRTIM_PERxR) (x = A to F)

Address offset: Block A: 0x114

Address offset: Block B: 0x214

Address offset: Block C: 0x314

Address offset: Block D: 0x414

Address offset: Block E: 0x514

Address offset: Block F: 0x614

Reset value: 0x0000 FFDF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PERx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PERx[15:0]**: Timer x period value

This register holds timer x period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The period value must be above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The maximum value is 0x0000 FFDF.

40.5.18 HRTIM timer x repetition register (HRTIM_REPxR) (x = A to F)

Address offset: Block A: 0x118

Address offset: Block B: 0x218

Address offset: Block C: 0x318

Address offset: Block D: 0x418

Address offset: Block E: 0x518

Address offset: Block F: 0x618

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **REPx[7:0]**: Timer x repetition period value

This register holds the repetition period value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

40.5.19 HRTIM timer x compare 1 register (HRTIM_CMP1xR) (x = A to F)

Address offset: Block A: 0x11C

Address offset: Block B: 0x21C

Address offset: Block C: 0x31C

Address offset: Block D: 0x41C

Address offset: Block E: 0x51C

Address offset: Block F: 0x61C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP1x[15:0]**: Timer x compare 1 value

This register holds the compare 1 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock, that is 0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

40.5.20 HRTIM timer x compare 1 compound register (HRTIM_CMP1CxR) (x = A to F)

Address offset: Block A: 0x120

Address offset: Block B: 0x220

Address offset: Block C: 0x320

Address offset: Block D: 0x420

Address offset: Block E: 0x520

Address offset: Block F: 0x620

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **REPx[7:0]**: Timer x repetition value aliased from HRTIM_REPx register
This bitfield is an alias from the REPx[7:0] bitfield in the HRTIMx_REPxR register.

Bits 15:0 **CMP1x[15:0]**: Timer x compare 1 value
This bitfield is an alias from the CMP1x[15:0] bitfield in the HRTIMx_CMP1xR register.

40.5.21 HRTIM timer x compare 2 register (HRTIM_CMP2xR) (x = A to F)

Address offset: Block A: 0x124

Address offset: Block B: 0x224

Address offset: Block C: 0x324

Address offset: Block D: 0x424

Address offset: Block E: 0x524

Address offset: Block F: 0x624

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP2x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP2x[15:0]**: Timer x compare 2 value

This register holds the compare 2 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock (0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...).

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

This register behaves as an auto-delayed compare register, if enabled with DELCMP2[1:0] bits in HRTIM_TIMxCR. In this case, the compare value must be strictly above or equal to 3 periods of the f_{HRTIM} clock.

40.5.22 HRTIM timer x compare 3 register (HRTIM_CMP3xR) (x = A to F)

Address offset: Block A: 0x128

Address offset: Block B: 0x228

Address offset: Block C: 0x328

Address offset: Block D: 0x428

Address offset: Block E: 0x528

Address offset: Block F: 0x628

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP3x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP3x[15:0]**: Timer x compare 3 value

This register holds the compare 3 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock (0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...).

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

40.5.23 HRTIM timer x compare 4 register (HRTIM_CMP4xR) (x = A to F)

Address offset: Block A: 0x12C

Address offset: Block B: 0x22C

Address offset: Block C: 0x32C

Address offset: Block D: 0x42C

Address offset: Block E: 0x52C

Address offset: Block F: 0x62C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP4x[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP4x[15:0]**: Timer x compare 4 value

This register holds the compare 4 value.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

The compare value must be either null or above or equal to 3 periods of the f_{HRTIM} clock (0x60 if CKPSC[2:0] = 0, 0x30 if CKPSC[2:0] = 1, 0x18 if CKPSC[2:0] = 2,...).

The null value is programmed following the use case described in *Section : Null duty cycle exception case*.

This register can behave as an auto-delayed compare register, if enabled with DELCMP4[1:0] bits in HRTIM_TIMxCR. In this case, the compare value must be strictly above or equal to 3 periods of the f_{HRTIM} clock.

40.5.24 HRTIM timer x capture 1 register (HRTIM_CPT1xR) (x = A to F)

Address offset: Block A: 0x130

Address offset: Block B: 0x230

Address offset: Block C: 0x330

Address offset: Block D: 0x430

Address offset: Block E: 0x530

Address offset: Block F: 0x630

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT1x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DIR**: Timer x capture 1 direction status
This register holds the counting direction value when the capture 1 event occurred:
0: timer is up-counting
1: timer is down-counting
In up-counting mode (UDM bit reset), the DIR bit is always read as 0.

Bits 15:0 **CPT1x[15:0]**: Timer x capture 1 value
This register holds the counter value when the capture 1 event occurred.

Note: In up/down mode (UDM bit set to 1), the capture value is referred to:
- counting reset, when up-counting
- the PER event when down counting
The DIR bit is used to discriminate the up-down phases when reading the captured value.

Note: This is a regular resolution register: for HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

40.5.25 HRTIM timer x capture 2 register (HRTIM_CPT2xR) (x = A to F)

Address offset: Block A: 0x134

Address offset: Block B: 0x234

Address offset: Block C: 0x334

Address offset: Block D: 0x434

Address offset: Block E: 0x534

Address offset: Block F: 0x634

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPT2x[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DIR**: Timer x capture 1 direction status
This register holds the counting direction value when the capture 1 event occurred:
0: timer is up-counting
1: timer is down-counting
In up-counting mode (UDM bit reset), the DIR bit is always read as 0.

Bits 15:0 **CPT2x[15:0]**: Timer x capture 2 value
This register holds the counter value when the capture 2 event occurred.

Note: In up/down mode (UDM bit set to 1), the capture value is referred to:
- counting reset, when up-counting
- the PER event when down counting

The DIR bit is used to discriminate the up-down phases when reading the captured value.

Note: This is a regular resolution register: for HR clock prescaling ratio below 32 (CKPSC[2:0] < 5), the least significant bits of the counter are not significant. They cannot be written and return 0 when read.

40.5.26 HRTIM timer x deadtime register (HRTIM_DTxR) (x = A to F)

Address offset: Block A: 0x138

Address offset: Block B: 0x238

Address offset: Block C: 0x338

Address offset: Block D: 0x438

Address offset: Block E: 0x538

Address offset: Block F: 0x638

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTFLKx	DTFSLKx	Res.	Res.	DTM[1:0]		SDTFx	DTFx[8:0]								
rwo	rwo			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTRLKx	DTRSLKx	Res.	DTPRSC[2:0]			SDTRx	DTRx[8:0]								
rwo	rwo		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DTFLKx**: Deadtime falling lock timer x

This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.

0: Deadtime falling value and sign is writable

1: Deadtime falling value and sign is read-only

Note: This bit is not preloaded

Bit 30 **DTFSLKx**: Deadtime falling sign lock timer x

This write-once bit prevents the sign of falling deadtime to be modified, if enabled.

0: Deadtime falling sign is writable

1: Deadtime falling sign is read-only

Note: This bit is not preloaded.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:26 **DTM[1:0]**: Deadtime mode

This bitfield defines the deadtime module behavior.

00: Deadtime complementary mode

01: Deadtime insertion mode

10: Deadtime insertion and disarm mode

11: Reserved. Writing 11 is discarded, and the DTM[1:0] value is maintained at its previous value.

Bit 25 **SDTFx**: Sign deadtime falling value timer x

This register determines whether the deadtime is positive (signals not overlapping) or negative (signals overlapping).

0: Positive deadtime on falling edge

1: Negative deadtime on falling edge

Bits 24:16 **DTFx[8:0]**: Deadtime falling value timer x

This register holds the value of the deadtime following a falling edge of reference PWM signal.

t_{DTF} = DTFx[8:0] x t_{DTG}

Bit 15 **DTRLKx**: Deadtime rising lock timer x
This write-once bit prevents the deadtime (sign and value) to be modified, if enabled.
0: Deadtime rising value and sign is writable
1: Deadtime rising value and sign is read-only
Note: This bit is not preloaded.

Bit 14 **DTRSLKx**: Deadtime rising sign lock timer x
This write-once bit prevents the sign of deadtime to be modified, if enabled.
0: Deadtime rising sign is writable
1: Deadtime rising sign is read-only
Note: This bit is not preloaded.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DTPRSC[2:0]**: Deadtime prescaler
This register holds the value of the deadtime clock prescaler determined by the following formula:
$t_{DTG} = (2^{(DTPRSC[2:0])}) \times (t_{HRTIM} / 8)$
(see *Table 425: Deadtime resolution and max absolute values*)
This bitfield is read-only as soon as any of the lock bit is enabled (DTFLKs, DTFSLKx, DTRLKx, DTRSLKx).

Bit 9 **SDTRx**: Sign deadtime rising value timer x
This register determines whether the deadtime is positive or negative (overlapping signals).
0: Positive deadtime on rising edge
1: Negative deadtime on rising edge

Bits 8:0 **DTRx[8:0]**: Deadtime rising value timer x
This register holds the value of the deadtime following a rising edge of reference PWM signal.
$t_{DTR} = DTRx[8:0] \times t_{DTG}$

40.5.27 HRTIM timer x output 1 set register (HRTIM_SETx1R) (x = A to F)

Address offset: Block A: 0x13C

Address offset: Block B: 0x23C

Address offset: Block C: 0x33C

Address offset: Block D: 0x43C

Address offset: Block E: 0x53C

Address offset: Block F: 0x63C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer from preload register to active register)
Register update event forces the output to its active state.

Bit 30 **EXTEVNTJ**: External event J
External event J forces the output to its active state.

Bit 29 **EXTEVNTI**: External event I
External event I forces the output to its active state.

Bit 28 **EXTEVNTH**: External event H
External event H forces the output to its active state.

Bit 27 **EXTEVNTG**: External event G
External event G forces the output to its active state.

Bit 26 **EXTEVNTF**: External event F
External event F forces the output to its active state.

Bit 25 **EXTEVNTE**: External event E
External event E forces the output to its active state.

Bit 24 **EXTEVNTD**: External event D
External event D forces the output to its active state.

Bit 23 **EXTEVNTC**: External event C
External event C forces the output to its active state.

Bit 22 **EXTEVNTB**: External event B
External event B forces the output to its active state.

Bit 21 **EXTEVNTA**: External event A
External event A forces the output to its active state.

Bit 20 **TIMEVNT9**: Timer event 9
Timers event 9 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 19 **TIMEVNT8**: Timer event 8
Timers event 8 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 18 **TIMEVNT7**: Timer event 7
Timers event 7 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 17 **TIMEVNT6**: Timer event 6
Timers event 6 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 16 **TIMEVNT5**: Timer event 5
Timers event 5 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 15 **TIMEVNT4**: Timer event 4
Timers event 4 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 14 **TIMEVNT3**: Timer event 3
Timers event 3 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 13 **TIMEVNT2**: Timer event 2
Timers event 2 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its active state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its active state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its active state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its active state.

Bit 7 **MSTPER**: Master period
The master timer counter period event (continuous mode), or the master timer counter reset (continous or single-shot mode), forces the output to its active state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its active state.

Bit 5 **CMP3**: Timer x compare 3
Timer A compare 3 event forces the output to its active state.

Bit 4 **CMP2**: Timer x compare 2
Timer A compare 2 event forces the output to its active state.

Bit 3 **CMP1**: Timer x compare 1
Timer A compare 1 event forces the output to its active state.

Bit 2 **PER**: Timer x period

Timer A period event forces the output to its active state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization

Timer A reset event coming solely from software or SYNC input forces the output to its active state.

Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SST**: Software set trigger

This bit forces the output to its active state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

40.5.28 HRTIM timer x output 1 reset register (HRTIM_RSTx1R) (x = A to F)

Address offset: Block A: 0x140

Address offset: Block B: 0x240

Address offset: Block C: 0x340

Address offset: Block D: 0x440

Address offset: Block E: 0x540

Address offset: Block F: 0x640

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer from preload register to active register)
Register update event forces the output to its inactive state.

Bit 30 **EXTEVNTJ**: External event J
External event J forces the output to its inactive state.

Bit 29 **EXTEVNTI**: External event I
External event I forces the output to its inactive state.

Bit 28 **EXTEVNTH**: External event H
External event H forces the output to its inactive state.

Bit 27 **EXTEVNTG**: External event G
External event G forces the output to its inactive state.

Bit 26 **EXTEVNTF**: External event F
External event F forces the output to its inactive state.

Bit 25 **EXTEVNTE**: External event E
External event E forces the output to its inactive state.

Bit 24 **EXTEVNTD**: External event D
External event D forces the output to its inactive state.

Bit 23 **EXTEVNTC**: External event C
External event C forces the output to its inactive state.

Bit 22 **EXTEVNTB**: External event B
External event B forces the output to its inactive state.

Bit 21 **EXTEVNTA**: External event A
External event A forces the output to its inactive state.

Bit 20 **TIMEVNT9**: Timer event 9
Timers event 9 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 19 **TIMEVNT8**: Timer event 8
Timers event 8 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 18 **TIMEVNT7**: Timer event 7
Timers event 7 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 17 **TIMEVNT6**: Timer event 6
Timers event 6 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 16 **TIMEVNT5**: Timer event 5
Timers event 5 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 15 **TIMEVNT4**: Timer event 4
Timers event 4 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 14 **TIMEVNT3**: Timer event 3
Timers event 3 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 13 **TIMEVNT2**: Timer event 2
Timers event 2 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its inactive state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its inactive state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its inactive state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its inactive state.

Bit 7 **MSTPER**: Master period
The master timer counter period event (continuous mode), or the master timer counter reset (continous or single-shot mode), forces the output to its inactive state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its inactive state.

Bit 5 **CMP3**: Timer x compare 3
Timer A compare 3 event forces the output to its inactive state.

Bit 4 **CMP2**: Timer x compare 2
Timer A compare 2 event forces the output to its inactive state.

Bit 3 **CMP1**: Timer x compare 1
Timer A compare 1 event forces the output to its inactive state.

Bit 2 **PER**: Timer x period

Timer A period event forces the output to its inactive state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization

Timer A reset event coming solely from software or SYNC input forces the output to its inactive state.

Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SRT**: Software reset trigger

This bit forces the output to its inactive state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

40.5.29 HRTIM timer x output 2 set register (HRTIM_SETx2R) (x = A to F)

Address offset: Block A: 0x144

Address offset: Block B: 0x244

Address offset: Block C: 0x344

Address offset: Block D: 0x444

Address offset: Block E: 0x544

Address offset: Block F: 0x644

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer from preload register to active register)

Register update event forces the output to its active state.

Bit 30 **EXTEVNTJ**: External event J

External event J forces the output to its active state.

Bit 29 **EXTEVNTI**: External event I

External event I forces the output to its active state.

Bit 28 **EXTEVNTH**: External event H

External event H forces the output to its active state.

Bit 27 **EXTEVNTG**: External event G

External event G forces the output to its active state.

Bit 26 **EXTEVNTF**: External event F

External event F forces the output to its active state.

Bit 25 **EXTEVNTE**: External event E
External event E forces the output to its active state.

Bit 24 **EXTEVNTD**: External event D
External event D forces the output to its active state.

Bit 23 **EXTEVNTC**: External event C
External event C forces the output to its active state.

Bit 22 **EXTEVNTB**: External event B
External event B forces the output to its active state.

Bit 21 **EXTEVNTA**: External event A
External event A forces the output to its active state.

Bit 20 **TIMEVNT9**: Timer event 9
Timers event 9 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 19 **TIMEVNT8**: Timer event 8
Timers event 8 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 18 **TIMEVNT7**: Timer event 7
Timers event 7 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 17 **TIMEVNT6**: Timer event 6
Timers event 6 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 16 **TIMEVNT5**: Timer event 5
Timers event 5 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 15 **TIMEVNT4**: Timer event 4
Timers event 4 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 14 **TIMEVNT3**: Timer event 3
Timers event 3 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 13 **TIMEVNT2**: Timer event 2
Timers event 2 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its active state (refer to *Table 422* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its active state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its active state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its active state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its active state.

Bit 7 **MSTPER**: Master period
The master timer counter period event (continuous mode), or the master timer counter reset (continous or single-shot mode), forces the output to its active state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its active state.

Bit 5 **CMP3**: Timer x compare 3
Timer A compare 3 event forces the output to its active state.

Bit 4 **CMP2**: Timer x compare 2
Timer A compare 2 event forces the output to its active state.

Bit 3 **CMP1**: Timer x compare 1
Timer A compare 1 event forces the output to its active state.

Bit 2 **PER**: Timer x period
Timer A period event forces the output to its active state.
Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization
Timer A reset event coming solely from software or SYNC input forces the output to its active state.
Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SST**: Software set trigger
This bit forces the output to its active state. This bit can only be set by software and is reset by hardware.
Note: This bit is not preloaded.

40.5.30 HRTIM timer x output 2 reset register (HRTIM_RSTx2R) (x = A to F)

Address offset: Block A: 0x148

Address offset: Block B: 0x248

Address offset: Block C: 0x348

Address offset: Block D: 0x448

Address offset: Block E: 0x548

Address offset: Block F: 0x648

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UPDATE**: Registers update (transfer from preload register to active register)
Register update event forces the output to its inactive state.

Bit 30 **EXTEVNTJ**: External event J
External event J forces the output to its inactive state.

Bit 29 **EXTEVNTI**: External event I
External event I forces the output to its inactive state.

Bit 28 **EXTEVNTH**: External event H
External event H forces the output to its inactive state.

Bit 27 **EXTEVNTG**: External event G
External event G forces the output to its inactive state.

Bit 26 **EXTEVNTF**: External event F
External event F forces the output to its inactive state.

Bit 25 **EXTEVNTE**: External event E
External event E forces the output to its inactive state.

Bit 24 **EXTEVNTD**: External event D
External event D forces the output to its inactive state.

Bit 23 **EXTEVNTC**: External event C
External event C forces the output to its inactive state.

Bit 22 **EXTEVNTB**: External event B
External event B forces the output to its inactive state.

Bit 21 **EXTEVNTA**: External event A
External event A forces the output to its inactive state.

Bit 20 **TIMEVNT9**: Timer event 9
Timers event 9 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 19 **TIMEVNT8**: Timer event 8
Timers event 8 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 18 **TIMEVNT7**: Timer event 7
Timers event 7 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 17 **TIMEVNT6**: Timer event 6
Timers event 6 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 16 **TIMEVNT5**: Timer event 5
Timers event 5 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 15 **TIMEVNT4**: Timer event 4
Timers event 4 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 14 **TIMEVNT3**: Timer event 3
Timers event 3 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 13 **TIMEVNT2**: Timer event 2
Timers event 2 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 12 **TIMEVNT1**:Timer event 1
Timers event 1 forces the output to its inactive state (refer to *Table 422* for timer events assignments).

Bit 11 **MSTCMP4**: Master compare 4
Master timer compare 4 event forces the output to its inactive state.

Bit 10 **MSTCMP3**: Master compare 3
Master timer compare 3 event forces the output to its inactive state.

Bit 9 **MSTCMP2**: Master compare 2
Master timer compare 2 event forces the output to its inactive state.

Bit 8 **MSTCMP1**: Master compare 1
Master timer compare 1 event forces the output to its inactive state.

Bit 7 **MSTPER**: Master period
The master timer counter period event (continuous mode), or the master timer counter reset (continous or single-shot mode), forces the output to its inactive state.

Bit 6 **CMP4**: Timer x compare 4
Timer A compare 4 event forces the output to its inactive state.

Bit 5 **CMP3**: Timer x compare 3
Timer A compare 3 event forces the output to its inactive state.

Bit 4 **CMP2**: Timer x compare 2
Timer A compare 2 event forces the output to its inactive state.

Bit 3 **CMP1**: Timer x compare 1
Timer A compare 1 event forces the output to its inactive state.

Bit 2 **PER**: Timer x period

Timer A period event forces the output to its inactive state.

Note: In up/down mode (UDM bit set to 1), the counter period event is defined as per the OUTROM[1:0] bit setting.

Bit 1 **RESYNC:** Timer A resynchronization

Timer A reset event coming solely from software or SYNC input forces the output to its inactive state.

Note: Other timer reset are not affecting the output when RESYNC=1.

Bit 0 **SRT**: Software reset trigger

This bit forces the output to its inactive state. This bit can only be set by software and is reset by hardware.

Note: This bit is not preloaded.

40.5.31 HRTIM timer x external event filtering register 1 (HRTIM_EEFxR1) (x = A to F)

Address offset: Block A: 0x14C

Address offset: Block B: 0x24C

Address offset: Block C: 0x34C

Address offset: Block D: 0x44C

Address offset: Block E: 0x54C

Address offset: Block F: 0x64C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEEFLTR[4:0]					EEELTCH	EEDFLTR[4:0]					EEDLTCH	EECFLTR[4:3]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EECFLTR[2:0]			EECLTCH	EEBFLTR[4:0]					EEBLTCH	EEAFLTR[4:0]					EEALTCH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **EEEFLTR[4:0]**: External event E filter
Refer to EEAFLTR[4:0] description.

Bit 24 **EEELTCH**: External event E latch
Refer to EEALTCH description

Bits 23:19 **EEDFLTR[4:0]**: External event D filter
Refer to EEAFLTR[4:0] description.

Bit 18 **EEDLTCH**: External event D latch
Refer to EEALTCH description

Bits 17:13 **EECFLTR[4:0]**: External event C filter
Refer to EEAFLTR[4:0] description.

Bit 12 **EECLTCH**: External event C latch
Refer to EEALTCH description

Bits 11:7 **EEBFLTR[4:0]**: External event B filter
Refer to EEAFLTR[4:0] description.

Bit 6 **EEBLTCH**: External event B latch
Refer to EEALTCH description.

Bits 5:1 **EEAFLTR[4:0]**: External event A filter

00000: No filtering

00001: Blanking from counter reset/roll-over to compare 1

00010: Blanking from counter reset/roll-over to compare 2 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): blanking from compare 1 to compare 2, only during the up-counting phase.

00011: Blanking from counter reset/roll-over to compare 3

00100: Blanking from counter reset/roll-over to compare 4 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): blanking from compare 3 to compare 4, only during the up-counting phase.

0XXXX: Blanking from another timing unit: TIMFLTRy source (see *Table 431* for details)
XXXX = 0101 → y = 1, ... , XXXX = 1100 → y = 8

01101: Windowing from counter reset/roll-over to compare 2 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 to compare 3, only during the up-counting phase.

01110: Windowing from counter reset/roll-over to compare 3 in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 to compare 3, only during the down-counting phase.

01111: Windowing from another timing unit: TIMWIN source (see *Table 432* for details) in up-counting mode (UDM bit reset)
In up-down counting mode (UDM bit set): windowing from compare 2 during the up-counting phase to compare 3 during the down-counting phase.

10000: Blanking from compare 1 to counter reset/roll-over (UDM bit reset).In up-down counting mode (UDM bit set): blanking from compare 1 during down-counting to compare 2 during the following up-counting phase.

10001: Blanking from compare 2 to counter reset/roll-over (UDM bit reset).In up-down counting mode (UDM bit set): blanking from CMP1 (down-counting) to CMP2 (down-counting).

10010: Blanking from compare 1 to compare 2 (UDM bit reset). In up-down counting mode (UDM bit set): blanking from CMP1 (up-counting) to CMP2 (down-counting.

10011: Blanking from compare 3 to compare 4 (UDM bit reset). In up-down counting mode (UDM bit set): blanking from CMP3 (up-counting) to CMP4 (down-counting).

10100: Windowing from compare 1 to counter reset/roll-over. In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

10101: Windowing from compare 2 to counter reset/roll-over. In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

10110: Windowing from compare 1 to compare 2 (UDM bit reset). In up-down counting mode (UDM bit set): windowing from CMP1 (up-counting) to CMP2 (down-counting).

10111: Windowing from compare 3 to compare 4 (UDM bit reset). In up-down counting mode (UDM bit set): windowing from compare 3 during down-counting to compare 4 during the following up-counting phase

11000: TIMWIN2 source (see *Table 432* for details). In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

11001: TIMWIN3 source (see *Table 432* for details)
In up-down counting mode (UDM bit set): windowing from compare 1 during up-counting to compare 2 during the down-counting.

11010: Blanking from Period Event to Compare 1 In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

11011: Blanking from Period Event to Compare 2 In up-down counting mode (UDM bit set), the reset/roll-over event is conditioned to the ROM[1:0] bitfield setting

11100: In push-pull mode only, blanking occurs from reset/roll-over to compare 1 when CPPSTAT = 0, and continuously when CPPSTAT = 1.

Note: Whenever a compare register is used for filtering, the value must be strictly above 0. This bitfield must not be modified once the counter is enabled (TxCEN bit set)

Bit 0 **EEALTCH**: External event A latch

0: Event A is ignored if it happens during a blank, or passed through during a window.

1: Event A is latched and delayed till the end of the blanking or windowing period.

Note: A timeout event is generated in window mode (EEAFLTR[4:0]=01101, 01110, 01111) if EEALTCH = 0, except if the external event is programmed in fast mode (EExFAST = 1).

This bitfield must not be modified once the counter is enabled (TxCEN bit set).

40.5.32 HRTIM timer x external event filtering register 2 (HRTIM_EEFxR2) (x = A to F)

Address offset: Block A: 0x150

Address offset: Block B: 0x250

Address offset: Block C: 0x350

Address offset: Block D: 0x450

Address offset: Block E: 0x550

Address offset: Block F: 0x650

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEJFLTR[4:0]					EEJLTCH	EEIFLTR[4:0]					EEILTCH	EEHFLTR[4:3]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EEHFLTR[2:0]			EEHLTCH	EEGFLTR[4:0]					EEGLTCH	EEFFLTR[4:0]					EEFLTCH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **EEJFLTR[4:0]**: External event J filter
Refer to EEAFLTR[4:0] description.

Bit 24 **EEJLTCH**: External event J latch
Refer to EEALTCH description.

Bits 23:19 **EEIFLTR[4:0]**: External event I filter
Refer to EEAFLTR[4:0] description.

Bit 18 **EEILTCH**: External event I latch
Refer to EEALTCH description.

Bits 17:13 **EEHFLTR[4:0]**: External event H filter
Refer to EEAFLTR[4:0] description.

Bit 12 **EEHLTCH**: External event H latch
Refer to EEALTCH description.

Bits 11:7 **EEGFLTR[4:0]**: External event G filter
Refer to EEAFLTR[4:0] description.

Bit 6 **EEGLTCH**: External event G latch
Refer to EEALTCH description.

Bits 5:1 **EEFFLTR[4:0]**: External event F filter
Refer to EEAFLTR[4:0] description.

Bit 0 **EEFLTCH**: External event F latch
Refer to EEALTCH description.

40.5.33 HRTIM timer A reset register (HRTIM_RSTAR)

Address offset: 0x154

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer A counter is reset upon timer F Compare 2 event.

Bit 30 **TIMECMP4**: Timer E compare 4
The timer A counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer A counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer A counter is reset upon timer E compare 1 event.

Bit 27 **TIMDCMP4**: Timer D compare 4
The timer A counter is reset upon timer D compare 4 event.

Bit 26 **TIMDCMP2**: Timer D compare 2
The timer A counter is reset upon timer D compare 2 event.

Bit 25 **TIMDCMP1**: Timer D compare 1
The timer A counter is reset upon timer D compare 1 event.

Bit 24 **TIMCCMP4**: Timer C compare 4
The timer A counter is reset upon timer C compare 4 event.

Bit 23 **TIMCCMP2**: Timer C compare 2
The timer A counter is reset upon timer C compare 2 event.

Bit 22 **TIMCCMP1**: Timer C compare 1
The timer A counter is reset upon timer C compare 1 event.

Bit 21 **TIMBCMP4**: Timer B compare 4
The timer A counter is reset upon timer B compare 4 event.

Bit 20 **TIMBCMP2**: Timer B compare 2
The timer A counter is reset upon timer B compare 2 event.

Bit 19 **TIMBCMP1**: Timer B compare 1
The timer A counter is reset upon timer B compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer A counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer A counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer A counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer A counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer A counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer A counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer A counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer A counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer A counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer A counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer A counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer A counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer A counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer A counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer A counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer A compare 4 reset
The timer A counter is reset upon timer A compare 4 event.

Bit 2 **CMP2**: Timer A compare 2 reset
The timer A counter is reset upon timer A compare 2 event.

Bit 1 **UPDT**: Timer A update reset
The timer A counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer A counter is reset upon timer F compare 1 event.

40.5.34 HRTIM timer B reset register (HRTIM_RSTBR)

Address offset: 0x254

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer B counter is reset upon timer F Compare 2 event.

Bit 30 **TIMECMP4**: Timer E compare 4
The timer B counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer B counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer B counter is reset upon timer E compare 1 event.

Bit 27 **TIMDCMP4**: Timer D compare 4
The timer B counter is reset upon timer D compare 4 event.

Bit 26 **TIMDCMP2**: Timer D compare 2
The timer B counter is reset upon timer D compare 2 event.

Bit 25 **TIMDCMP1**: Timer D compare 1
The timer B counter is reset upon timer D compare 1 event.

Bit 24 **TIMCCMP4**: Timer C compare 4
The timer B counter is reset upon timer C compare 4 event.

Bit 23 **TIMCCMP2**: Timer C compare 2
The timer B counter is reset upon timer C compare 2 event.

Bit 22 **TIMCCMP1**: Timer C compare 1
The timer B counter is reset upon timer C compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer B counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer B counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer B counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer B counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer B counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer B counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer B counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer B counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer B counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer B counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer B counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer B counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer B counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer B counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer B counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer B counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer B counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer B counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer B compare 4 reset
The timer B counter is reset upon timer B compare 4 event.

Bit 2 **CMP2**: Timer B compare 2 reset
The timer B counter is reset upon timer B compare 2 event.

Bit 1 **UPDT**: Timer B update reset
The timer B counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer B counter is reset upon timer F compare 1 event.

40.5.35 HRTIM timer C reset register (HRTIM_RSTCR)

Address offset: 0x354

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer C counter is reset upon timer F Compare 2 event.

Bit 30 **TIMECMP4**: Timer E compare 4
The timer C counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer C counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer C counter is reset upon timer E compare 1 event.

Bit 27 **TIMDCMP4**: Timer D compare 4
The timer C counter is reset upon timer D compare 4 event.

Bit 26 **TIMDCMP2**: Timer D compare 2
The timer C counter is reset upon timer D compare 2 event.

Bit 25 **TIMDCMP1**: Timer D compare 1
The timer C counter is reset upon timer D compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4
The timer C counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2
The timer C counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1
The timer C counter is reset upon timer B compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer C counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer C counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer C counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer C counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer C counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer C counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer C counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer C counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer C counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer Ccounter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer C counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer C counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer C counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer C counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer C counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer C counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer C counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer C counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer C compare 4 reset
The timer C counter is reset upon timer C compare 4 event.

Bit 2 **CMP2**: Timer C compare 2 reset
The timer C counter is reset upon timer C compare 2 event.

Bit 1 **UPDT**: Timer C update reset
The timer C counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer C counter is reset upon timer F compare 1 event.

40.5.36 HRTIM timer D reset register (HRTIM_RSTDR)

Address offset: 0x454

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer D counter is reset upon timer F Compare 2 event.

Bit 30 **TIMECMP4**: Timer E compare 4
The timer D counter is reset upon timer E compare 4 event.

Bit 29 **TIMECMP2**: Timer E compare 2
The timer D counter is reset upon timer E compare 2 event.

Bit 28 **TIMECMP1**: Timer E compare 1
The timer D counter is reset upon timer E compare 1 event.

Bit 27 **TIMCCMP4**: Timer C compare 4
The timer D counter is reset upon timer C compare 4 event.

Bit 26 **TIMCCMP2**: Timer C compare 2
The timer D counter is reset upon timer C compare 2 event.

Bit 25 **TIMCCMP1**: Timer C compare 1
The timer D counter is reset upon timer C compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4
The timer D counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2
The timer D counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1
The timer D counter is reset upon timer B compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer D counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer D counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer D counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer A counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer A counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer A counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer A counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer A counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer A counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer A counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer A counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer A counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer A counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer D counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer D counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer D counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer D counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer D counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer D compare 4 reset
The timer D counter is reset upon timer D compare 4 event.

Bit 2 **CMP2**: Timer D compare 2 reset
The timer D counter is reset upon timer D compare 2 event.

Bit 1 **UPDT**: Timer D update reset
The timer D counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer D counter is reset upon timer F compare 1 event.

40.5.37 HRTIM timer E reset register (HRTIM_RSTER)

Address offset: 0x554

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMFCMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMFCMP2**: Timer F compare 2
The timer E counter is reset upon timer F Compare 2 event.

Bit 30 **TIMDCMP4**: Timer D compare 4
The timer E counter is reset upon timer D compare 4 event.

Bit 29 **TIMDCMP2**: Timer D compare 2
The timer E counter is reset upon timer D compare 2 event.

Bit 28 **TIMDCMP1**: Timer D compare 1
The timer E counter is reset upon timer D compare 1 event.

Bit 27 **TIMCCMP4**: Timer C compare 4
The timer E counter is reset upon timer C compare 4 event.

Bit 26 **TIMCCMP2**: Timer C compare 2
The timer E counter is reset upon timer C compare 2 event.

Bit 25 **TIMCCMP1**: Timer C compare 1
The timer E counter is reset upon timer C compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4
The timer E counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2
The timer E counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1
The timer E counter is reset upon timer B compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer E counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer E counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer E counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer E counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer E counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer E counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer E counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer E counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer E counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer E counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer E counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer E counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer E counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer E counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer E counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer E counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer E counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer E counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer E compare 4 reset
The timer E counter is reset upon timer E compare 4 event.

Bit 2 **CMP2**: Timer E compare 2 reset
The timer E counter is reset upon timer E compare 2 event.

Bit 1 **UPDT**: Timer E update reset
The timer E counter is reset upon update event.

Bit 0 **TIMFCMP1**: Timer F compare 1
The timer E counter is reset upon timer F compare 1 event.

40.5.38 HRTIM timer F reset register (HRTIM_RSTFR)

Address offset: 0x654

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMECMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMECMP1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMECMP2**: Timer E compare 2
The timer F counter is reset upon timer E Compare 2 event.

Bit 30 **TIMDCMP4**: Timer D compare 4
The timer F counter is reset upon timer D compare 4 event.

Bit 29 **TIMDCMP2**: Timer D compare 2
The timer F counter is reset upon timer D compare 2 event.

Bit 28 **TIMDCMP1**: Timer D compare 1
The timer F counter is reset upon timer D compare 1 event.

Bit 27 **TIMCCMP4**: Timer C compare 4
The timer F counter is reset upon timer C compare 4 event.

Bit 26 **TIMCCMP2**: Timer C compare 2
The timer F counter is reset upon timer C compare 2 event.

Bit 25 **TIMCCMP1**: Timer C compare 1
The timer F counter is reset upon timer C compare 1 event.

Bit 24 **TIMBCMP4**: Timer B compare 4
The timer F counter is reset upon timer B compare 4 event.

Bit 23 **TIMBCMP2**: Timer B compare 2
The timer F counter is reset upon timer B compare 2 event.

Bit 22 **TIMBCMP1**: Timer B compare 1
The timer F counter is reset upon timer B compare 1 event.

Bit 21 **TIMACMP4**: Timer A compare 4
The timer F counter is reset upon timer A compare 4 event.

Bit 20 **TIMACMP2**: Timer A compare 2
The timer F counter is reset upon timer A compare 2 event.

Bit 19 **TIMACMP1**: Timer A compare 1
The timer F counter is reset upon timer A compare 1 event.

Bit 18 **EXTEVNTJ**: External event J
The timer F counter is reset upon external event J.

Bit 17 **EXTEVNTI**: External event I
The timer F counter is reset upon external event I.

Bit 16 **EXTEVNTH**: External event H
The timer F counter is reset upon external event H.

Bit 15 **EXTEVNTG**: External event G
The timer F counter is reset upon external event G.

Bit 14 **EXTEVNTF**: External event F
The timer F counter is reset upon external event F.

Bit 13 **EXTEVNTE**: External event E
The timer F counter is reset upon external event E.

Bit 12 **EXTEVNTD**: External event D
The timer F counter is reset upon external event D.

Bit 11 **EXTEVNTC**: External event C
The timer F counter is reset upon external event C.

Bit 10 **EXTEVNTB**: External event B
The timer F counter is reset upon external event B.

Bit 9 **EXTEVNTA**: External event A
The timer F counter is reset upon external event A.

Bit 8 **MSTCMP4**: Master compare 4
The timer F counter is reset upon master timer compare 4 event.

Bit 7 **MSTCMP3**: Master compare 3
The timer F counter is reset upon master timer compare 3 event.

Bit 6 **MSTCMP2**: Master compare 2
The timer F counter is reset upon master timer compare 2 event.

Bit 5 **MSTCMP1**: Master compare 1
The timer F counter is reset upon master timer compare 1 event.

Bit 4 **MSTPER**: Master timer period
The timer F counter is reset upon master timer period event.

Bit 3 **CMP4**: Timer F compare 4 reset
The timer F counter is reset upon timer F compare 4 event.

Bit 2 **CMP2**: Timer F compare 2 reset
The timer F counter is reset upon timer F compare 2 event.

Bit 1 **UPDT**: Timer F update reset
The timer F counter is reset upon update event.

Bit 0 **TIMECMP1**: Timer E compare 1
The timer F counter is reset upon timer E compare 1 event.

40.5.39 HRTIM timer x chopper register (HRTIM_CHPxR) (x = A to F)

Address offset: Block A: 0x158

Address offset: Block B: 0x258

Address offset: Block C: 0x358

Address offset: Block D: 0x458

Address offset: Block E: 0x558

Address offset: Block F: 0x658

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	STRTPW[3:0]				CARDTY[2:0]			CARFRQ[3:0]			
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:7 **STRTPW[3:0]**: Timer x start pulsewidth

This register defines the initial pulsewidth following a rising edge on output signal.
This bitfield cannot be modified when one of the CHPx bits is set.
t_{1STPW} = (STRPW[3:0]+1) x 16 x t_{HRTIM}.
0000: 57 ns (1/17.5 MHz)
...
1111: 910 ns (16/17.5 MHz)

Bits 6:4 **CARDTY[2:0]**: Timer x chopper duty cycle value

This register defines the duty cycle of the carrier signal. This bitfield cannot be modified when one of the CHPx bits is set.
000: 0/8 (i.e. only 1st pulse is present)
...
111: 7/8

Bits 3:0 **CARFRQ[3:0]**: Timer x carrier frequency value

This register defines the carrier frequency F_{CHPFRQ} = f_{HRTIM} / (16 x (CARFRQ[3:0]+1)).
This bitfield cannot be modified when one of the CHPx bits is set.
0000: 17.5 MHz (f_{HRTIM}/ 16)
...
1111: 1.09 MHz (f_{HRTIM} / 256)

40.5.40 HRTIM timer A capture 1 control register (HRTIM_CPT1ACR)

Address offset: 0x15C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TFCMP2	TFCMP1	TF1RST	TF1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TFCMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 14 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 13 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 12 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

40.5.41 HRTIM timer B capture 1 control register (HRTIM_CPT1BCR)

Address offset: 0x25C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TF1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 18 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 17 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 16 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

40.5.42 HRTIM timer C capture 1 control register (HRTIM_CPT1CCR)

Address offset: 0x35C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 22 **TFCMP1**:Timer F compare 1
Refer to TACMP1 description.

Bit 21 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 20 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

40.5.43 HRTIM timer D capture 1 control register (HRTIM_CPT1DCR)

Address offset: 0x45C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 26 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 25 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 24 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

40.5.44 HRTIM timer E capture 1 control register (HRTIM_CPT1ECR)

Address offset: 0x55C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 30 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 29 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 28 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

40.5.45 HRTIM timer F capture 1 control register (HRTIM_CPT1FCR)

Address offset: 0x65C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 1

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 1

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 1 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 1 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 1

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 1

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 1 by software. This bit is set only, reset by hardware.

40.5.46 HRTIM timer A capture 2 control register (HRTIM_CPT2ACR)

Address offset: 0x160

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TFCMP2	TFCMP1	TF1RST	TF1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TFCMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 14 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 13 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 12 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

40.5.47 HRTIM timer B capture 2 control register (HRTIM_CPT2BCR)

Address offset: 0x260

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TF1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 18 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 17 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 16 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

40.5.48 HRTIM timer C capture 2 control register (HRTIM_CPT2CCR)

Address offset: 0x360

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 22 **TFCMP1**:Timer F compare 1
Refer to TACMP1 description.

Bit 21 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 20 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

40.5.49 HRTIM timer D capture 2 control register (HRTIM_CPT2DCR)

Address offset: 0x460

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 26 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 25 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 24 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

40.5.50 HRTIM timer E capture 2 control register (HRTIM_CPT2ECR)

Address offset: 0x560

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TFCMP2**: Timer F compare 2
Refer to TACMP2 description.

Bit 30 **TFCMP1**: Timer F compare 1
Refer to TACMP1 description.

Bit 29 **TF1RST**: Timer F output 1 reset
Refer to TA1RST description.

Bit 28 **TF1SET**: Timer F output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

40.5.51 HRTIM timer F capture 2 control register (HRTIM_CPT2FCR)

Address offset: 0x660

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCPT	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TECMP2**: Timer E compare 2
Refer to TACMP2 description.

Bit 30 **TECMP1**: Timer E compare 1
Refer to TACMP1 description.

Bit 29 **TE1RST**: Timer E output 1 reset
Refer to TA1RST description.

Bit 28 **TE1SET**: Timer E output 1 set
Refer to TA1SET description.

Bit 27 **TDCMP2**: Timer D compare 2
Refer to TACMP2 description.

Bit 26 **TDCMP1**:Timer D compare 1
Refer to TACMP1 description.

Bit 25 **TD1RST**: Timer D output 1 reset
Refer to TA1RST description.

Bit 24 **TD1SET**: Timer D output 1 set
Refer to TA1SET description.

Bit 23 **TCCMP2**: Timer C compare 2
Refer to TACMP2 description.

Bit 22 **TCCMP1**:Timer C compare 1
Refer to TACMP1 description.

Bit 21 **TC1RST**: Timer C output 1 reset
Refer to TA1RST description.

Bit 20 **TC1SET**: Timer C output 1 set
Refer to TA1SET description.

Bit 19 **TBCMP2**: Timer B compare 2
Refer to TACMP2 description.

Bit 18 **TBCMP1**: Timer B compare 1
Refer to TACMP1 description.

Bit 17 **TB1RST**: Timer B output 1 reset
Refer to TA1RST description.

Bit 16 **TB1SET**: Timer B output 1 set
Refer to TA1SET description.

Bit 15 **TACMP2**: Timer A compare 2
0: No action
1: Timer A compare 2 triggers capture 2

Bit 14 **TACMP1**: Timer A compare 1
0: No action
1: Timer A compare 1 triggers capture 2

Bit 13 **TA1RST**: Timer B output 1 reset
0: No action
1: Capture 2 is triggered by TA1 output active to inactive transition

Bit 12 **TA1SET**: Timer B output 1 set
0: No action
1: Capture 2 is triggered by TA1 output inactive to active transition

Bit 11 **EXEVJCPT**: External event J capture
Refer to EXEVACPT description

Bit 10 **EXEVICPT**: External event I capture
Refer to EXEVACPT description.

Bit 9 **EXEVHCPT**: External event H capture
Refer to EXEVACPT description.

Bit 8 **EXEVGCPT**: External event G capture
Refer to EXEVACPT description.

Bit 7 **EXEVFCPT**: External event F capture
Refer to EXEVACPT description.

Bit 6 **EXEVECPT**: External event E capture
Refer to EXEVACPT description.

Bit 5 **EXEVDCPT**: External event D capture
Refer to EXEVACPT description.

Bit 4 **EXEVCCPT**: External event C capture
Refer to EXEVACPT description.

Bit 3 **EXEVBCPT**: External event B capture
Refer to EXEVACPT description.

Bit 2 **EXEVACPT**: External event A capture
0: No action
1: The external event A triggers the capture 2

Bit 1 **UPDCPT**: Update capture
0: No action
1: The update event triggers the capture 2

Bit 0 **SWCPT**: Software capture
0: No action
1: This bit forces the capture 2 by software. This bit is set only, reset by hardware.

40.5.52 HRTIM timer x output register (HRTIM_OUTxR) (x = A to F)

Address offset: Block A: 0x164

Address offset: Block B: 0x264

Address offset: Block C: 0x364

Address offset: Block D: 0x464

Address offset: Block E: 0x564

Address offset: Block F: 0x664

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM 2	POL2	Res.
								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BIAR	Res.	DLYPRT[2:0]			DLYPR TEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM 1	POL1	Res.
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DIDL2**: Output 2 deadtime upon burst mode Idle entry

This bit delays the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering in idle state during a burst mode operation.

0: The programmed Idle state is applied immediately to the output 2

1: Deadtime (inactive level) is inserted on output 2 before entering the idle mode. The deadtime value is set by DTFx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 is set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 22 **CHP2**: Output 2 chopper enable

This bit enables the chopper on output 2.

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 21:20 **FAULT2[1:0]**: Output 2 fault state

These bits select the output 2 state after a fault event.

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 19 **IDLES2**: Output 2 idle state

This bit selects the output 2 idle state.

0: Inactive

1: Active

Note: This parameter must be set prior to have the HRTIM controlling the outputs.

Bit 18 **IDLEM2**: Output 2 idle mode
This bit selects the output 2 idle mode.
0: No action: the output is not affected by the burst mode operation.
1: The output is in idle state when requested by the burst mode controller.
Note: This bit is preloaded and is changed during run-time, but must not be changed while the burst mode is active.

Bit 17 **POL2**: Output 2 polarity
This bit selects the output 2 polarity.
0: Positive polarity (output active high)
1: Negative polarity (output active low)
Note: This parameter cannot be changed once the timer x is enabled.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **BIAR**: Balanced Idle Automatic Resume
This bit selects is the outputs are automatically re-enabled after a balanced idle event.
This bit is only significant if DLYPRT[2:0] = 011 or 111, it is ignored otherwise.
0: Disabled
1: Enabled
Note: This parameter cannot be changed once the timer x is enabled.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **DLYPRT[2:0]**: Delayed protection
These bits define the source and outputs on which the delayed protection schemes are applied.
In HRTIM_OUTAR, HRTIM_OUTBR, HRTIM_OUTCR:
000: Output 1 delayed idle on external event F
001: Output 2 delayed idle on external event F
010: Output 1 and output 2 delayed idle on external event F
011: Balanced idle on external event F
100: Output 1 delayed idle on external event G
101: Output 2 delayed idle on external event G
110: Output 1 and output 2 delayed idle on external event G
111: Balanced idle on external event G
In HRTIM_OUTDR, HRTIM_OUTER, HRTIM_OUTFR:
000: Output 1 delayed idle on external event H
001: Output 2 delayed idle on external event H
010: Output 1 and output 2 delayed idle on external event H
011: Balanced idle on external event H
100: Output 1 delayed idle on external event I
101: Output 2 delayed idle on external event I
110: Output 1 and output 2 delayed idle on external event I
111: Balanced idle on external event I
Note: This bitfield must not be modified once the delayed protection is enabled (DLYPRTEN bit set).

Bit 9 **DLYPRTEN**: Delayed protection enable
This bit enables the delayed protection scheme.
0: No action
1: Delayed protection is enabled, as per DLYPRT[2:0] bits
Note: This parameter cannot be changed once the timer x is enabled (TxEN bit set).

Bit 8 **DTEN**: Deadtime enable

This bit enables the deadtime insertion on output 1 and output 2.

0: Output 1 and output 2 signals are independent.

1: Deadtime is inserted between output 1 and output 2 (reference signal is output 1 signal generator)

Note: This parameter cannot be changed if the timer outputs are enabled and set/reset by another timer.

Bit 7 **DIDL1**: Output 1 deadtime upon burst mode idle entry

This bit delays the idle mode entry by forcing a deadtime insertion before switching the outputs to their idle state. This setting only applies when entering the idle state during a burst mode operation.

0: The programmed idle state is applied immediately to the output 1

1: Deadtime (inactive level) is inserted on output 1 before entering the idle mode. The deadtime value is set by DTRx[8:0].

Note: This parameter cannot be changed once the timer x is enabled.

DIDL=1 is set only if one of the outputs is active during the burst mode (IDLES=1), and with positive deadtimes (SDTR/SDTF set to 0).

Bit 6 **CHP1**: Output 1 chopper enable

This bit enables the chopper on output 1.

0: Output signal is not altered

1: Output signal is chopped by a carrier signal

Note: This parameter cannot be changed once the timer x is enabled.

Bits 5:4 **FAULT1[1:0]**: Output 1 fault state

These bits select the output 1 state after a fault event

00: No action: the output is not affected by the fault input and stays in run mode.

01: Active

10: Inactive

11: High-Z

Note: This parameter cannot be changed once the timer x is enabled (TxCEN bit set), if FLTENx bit is set or if the output is in FAULT state.

Bit 3 **IDLES1**: Output 1 Idle State

This bit selects the output 1 idle state.

0: Inactive

1: Active

Note: This parameter must be set prior to HRTIM controlling the outputs.

Bit 2 **IDLEM1**: Output 1 Idle mode

This bit selects the output 1 idle mode.

0: No action: the output is not affected by the burst mode operation.

1: The output is in idle state when requested by the burst mode controller.

Note: This bit is preloaded and is changed during runtime, but must not be changed while burst mode is active.

Bit 1 **POL1**: Output 1 polarity

This bit selects the output 1 polarity.

0: Positive polarity (output active high)

1: Negative polarity (output active low)

Note: This parameter cannot be changed once the timer x is enabled.

Bit 0 Reserved, must be kept at reset value.

40.5.53 HRTIM timer x fault register (HRTIM_FLTxR) (x = A to F)

Address offset: Block A: 0x168

Address offset: Block B: 0x268

Address offset: Block C: 0x368

Address offset: Block D: 0x468

Address offset: Block E: 0x568

Address offset: Block F: 0x668

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT7A8EN	FLT6A7EN	FLT5A6EN	FLT4A5EN	FLT3A4EN	FLT2A3EN	FLT1A2EN
rwo									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8EN	FLT7EN	FLT6EN	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLTLCK**: Fault sources lock
0: FLT1EN..FLT6EN bits are read/write
1: FLT1EN..FLT6EN bits are read only
The FLTLCK bit is write-once. Once it has been set, it cannot be modified till the next system reset.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **FLT7A8EN**: Fault 7 AND Fault 8 enable
0: Fault 7 AND Fault 8 ignored
1: HRTIM outputs disabled when both Fault 7 AND Fault 8 inputs are active

Bit 21 **FLT6A7EN**: Fault 6 AND Fault 7 enable
0: Fault 6 AND Fault 7 ignored
1: HRTIM outputs disabled when both Fault 6 AND Fault 7 inputs are active

Bit 20 **FLT5A6EN**: Fault 5 AND Fault 6 enable
0: Fault 5 AND Fault 6 ignored
1: HRTIM outputs disabled when both Fault 5 AND Fault 6 inputs are active

Bit 19 **FLT4A5EN**: Fault 4 AND Fault 5 enable
0: Fault 4 AND Fault 5 ignored
1: HRTIM outputs disabled when both Fault 4 AND Fault 5 inputs are active

Bit 18 **FLT3A4EN**: Fault 3 AND Fault 4 enable
0: Fault 3 AND Fault 4 ignored
1: HRTIM outputs disabled when both Fault 3 AND Fault 4 inputs are active

Bit 17 **FLT2A3EN**: Fault 2 AND Fault 3 enable
0: Fault 2 AND Fault 3 ignored
1: HRTIM outputs disabled when both Fault 2 AND Fault 3 inputs are active

Bit 16 **FLT1A2EN**: Fault 1 AND Fault 2 enable
0: Fault 1 AND Fault 2 ignored
1: HRTIM outputs disabled when both Fault 1 AND Fault 2 inputs are active

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **FLT8EN**: Fault 8 enable
0: Fault 8 input ignored
1: Fault 8 input is active and disables HRTIM outputs

Bit 6 **FLT7EN**: Fault 7 enable
0: Fault 7 input ignored
1: Fault 7 input is active and disables HRTIM outputs

Bit 5 **FLT6EN**: Fault 6 enable
0: Fault 6 input ignored
1: Fault 6 input is active and disables HRTIM outputs

Bit 4 **FLT5EN**: Fault 5 enable
0: Fault 5 input ignored
1: Fault 5 input is active and disables HRTIM outputs

Bit 3 **FLT4EN**: Fault 4 enable
0: Fault 4 input ignored
1: Fault 4 input is active and disables HRTIM outputs

Bit 2 **FLT3EN**: Fault 3 enable
0: Fault 3 input ignored
1: Fault 3 input is active and disables HRTIM outputs

Bit 1 **FLT2EN**: Fault 2 enable
0: Fault 2 input ignored
1: Fault 2 input is active and disables HRTIM outputs

Bit 0 **FLT1EN**: Fault 1 enable
0: Fault 1 input ignored
1: Fault 1 input is active and disables HRTIM outputs

40.5.54 HRTIM timer x control register 2 (HRTIM_TIMxCR2) (x = A to F)

Address offset: Block A: 0x16C

Address offset: Block B: 0x26C

Address offset: Block C: 0x36C

Address offset: Block D: 0x46C

Address offset: Block E: 0x56C

Address offset: Block F: 0x66C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DCDCOE	COM	COR	COE	ADCTRGCH2	ADCTRGCH1	TRGHLFT	TRGHLF	Res.	Res.	GTCMP3	GTCMP1
				rw	rw	w	rw	rw	rw	rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEROM[1:0]		BMROM[1:0]		ADROM[1:0]		OUTROM[1:0]		ROM[1:0]		Res.	UDM	Res.	DCDR	DCDS	DCDE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **DCDCOE**: Dual channel DAC crossbar override enable

0: Dual Channel DAC Crossbar Override disabled

1: Dual Channel DAC Crossbar Override enabled

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 26 **COM**: Crossbar override mode

This bit defines if the crossbar override mode is entered only once (latched mode), in which case it is necessary to rearm it to resume standard crossbar operation, using the COR bit.

In cycle-by-cycle mode, the standard crossbar operation is automatically resumed on the next following reset/roll-over event, once the triggering condition have disappeared. This mode is only applicable in force on event mode (COFMx[2:0] = 100).

0: Cycle-by-cycle mode

1: latched mode

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 25 **COR**: Crossbar override rearm

This bit is set by software and automatically reset by hardware. It resume standard crossbar operation and rearms the crossbar override mode. The rearm operation is only effective if the crossbar override triggering condition have disappeared. If the trigger is still active, the rearm request is ignored and the crossbar override mode is maintained active.

Bit 24 **COE**: Crossbar override enable

0: Crossbar override disabled

1: Crossbar override enabled

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 23 **ADCTRGCH2**: ADC trigger for channel 2

This bit is significant only in push-pull mode. It gives the possibility to have ADC triggers issued only when Channel 2 output is active

0: ADC triggers generated for both Channel 1 and Channel 2 active phases

1: ADC triggers generated when Channel 2 is active (in push-pull mode)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 22 **ADCTRGCH1**: ADC trigger for channel 1

This bit is significant only in push-pull mode. It gives the possibility to have ADC triggers issued only when Channel 1 output is active

0: ADC triggers generated for both Channel 1 and Channel 2 active phases

1: ADC triggers generated when Channel 1 is active (in push-pull mode)

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 21 **TRGHLFT**: Triggered-half timeout mode

This bit defines if a timeout event must be generated when the synchronization disappear (i.e. no more EEV event

0: No timeout event generated

1: timeout event generated. When the counter matches the PER value, the CMP2 register takes PER/2 value.

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 20 **TRGHLF**: Triggered-half mode

This bit defines whether the compare 2 register is behaving in standard mode (compare match issued as soon as counter equal compare), or in triggered-half mode (see *Section : Triggered-half mode*).

0: CMP2 register is written by the user only (standard compare mode)

1: CMP2 value is set by hardware as soon as a capture 1 event occurs. It is loaded with the (capture 1 divided by 2) value. The initial value can be written by the user (as long as TRGHLF is reset), but is ignored once the first capture has been triggered (the preload mechanism is disabled for CMP2 when the TRGHLF bit is set).

Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **GTCMP3**: Greater than compare 3 PWM mode

This bit defines the compare 3 operating mode.

0: The compare 3 event is generated when the counter is equal to the compare value (compare match mode)

1: The compare 3 event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

Bit 16 **GTCMP1**: Greater than compare 1 PWM mode

This bit defines the compare 1 operating mode.

0: The compare 1 event is generated when the counter is equal to the compare value (compare match mode)

1: The compare 1 event is generated when the counter is greater than the compare value. If the compare value is changed on-the-fly, the new compare value is compared with the current counter value and an output SET or RESET can be generated.

Bits 15:14 **FEROM[1:0]**: Fault and event roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the Roll-over event used by the fault and event counters.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 13:12 **BMROM[1:0]**: Burst mode roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event used in the burst mode controller, as clock as burst mode trigger.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This parameter cannot be changed once the timer is operating (TxEN bit set).

Bits 11:10 **ADROM[1:0]**: ADC roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event which triggers the ADC.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 9:8 **OUTROM[1:0]**: Output roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event which sets and/or resets the ouputs, as per HRTIM_SETxyR and HRTIM_RSTxyR settings.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bits 7:6 **ROM[1:0]**: Roll-over mode

This bit defines when the roll-over is generated, in up-down counting mode. It only concerns the roll-over event with the following destinations: update trigger (to transfer content from preload to active registers), IRQ and DMA requests, repetition counter decrement and external event filtering.

00: Event generated when the counter is equal to 0 or to HRTIM_PERxR value

01: Event generated when the counter is equal to 0

10: Event generated when the counter is equal to HRTIM_PERxR

11: Reserved

Note: This setting only applies when the UDM bit is set. It is not significant otherwise.

Note: This bitfield cannot be changed once the timer is operating (TxEN bit set).

Bit 5 Reserved, must be kept at reset value.

Bit 4 **UDM**: Up-Down mode
This bit defines if the counter is operating in up or up-down counting mode.
0: The counter is operating in up-counting mode
1: The counter is operating in up-down counting mode
Note: This bit cannot be changed once the timer is operating (TxEN bit set).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCDR**: Dual channel DAC reset trigger
This bit defines when the hrtim_dac_reset_trgx trigger is generated.
0: The trigger is generated on counter reset or roll-over event
1: The trigger is generated on output 1 set event
Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

Bit 1 **DCDS**: Dual channel DAC Step trigger
This bit defines when the hrtim_dac_step_trgx trigger is generated.
0: The trigger is generated on compare 2 event
1: The trigger is generated on output 1 reset event
Note: The DCDR bit is not significant when the DCDE bit is reset (Dual channel DAC trigger disabled).

Bit 0 **DCDE**: Dual channel DAC trigger enable
This bit enables the dual channel DAC triggering mechanism.
0: Dual channel DAC trigger disabled
1: Dual channel DAC trigger enabled
Note: This bit cannot be changed once the timer is operating (TxEN bit set).

40.5.55 HRTIM timer x external event filtering register 3 (HRTIM_TIMxEEFR3) (x = A to F)

Address offset: Block A: 0x170

Address offset: Block B: 0x270

Address offset: Block C: 0x370

Address offset: Block D: 0x470

Address offset: Block E: 0x570

Address offset: Block F: 0x670

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEVZCNT[5:0]						EEVZSEL[3:0]				Res.	EEVZRSTM	EEVZCRES	EEVZCE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EEVYCNT[5:0]						EEVYSEL[3:0]				Res.	EEVYRSTM	EEVYCRES	EEVYCE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **EEVZCNT[5:0]**: External event Z counter

This bitfield selects the external event Z counter threshold. An event is considered valid when the number of events is equal to the (EEVZCNT[5:0]+1) value.

Bits 23:20 **EEVZSEL[3:0]**: External event Z selection

This bit selects the external event B source.

0: External event A is used as external event Z source

1: External event B is used as external event Z source

...

9: External event J is used as external event Z source

Others: Reserved

Bit 19 Reserved, must be kept at reset value.

Bit 18 **EEVZRSTM**: External event Z reset mode

This bit selects the external event Z counter reset mode

0: External event counter Z is reset on each reset / roll-over event

1: External event counter Z is reset on each reset / roll-over event only if no event occurs during last counting period

Bit 17 **EEVZCRES**: External event Z counter reset

This bit resets the external event Z counter. It is set by software and reset by hardware.

0: No action

1: External event counter Z counter is reset

Bit 16 **EEVZCE**: External event Z counter enable

This bit enables the external event Z counter.

0: External event Z counter disabled

1: External event Z counter enabled

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **EEVYCNT[5:0]**: External event Y counter
This bitfield selects the external event Y counter threshold. An event is considered valid when the number of events is equal to the (EEVYCNT[5:0]+1) value.

Bits 7:4 **EEVYSEL[3:0]**: External event Y Selection
This bit selects the external event Y source.
0: External event A is used as external event Y source
1: External event B is used as external event Y source
...
9: External event J is used as external event Y source
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EEVYRSTM**: External event Y reset mode
This bit selects the external event Y counter reset mode.
0: External event counter Y is reset on each reset / roll-over event
1: External event counter Y is reset on each reset / roll-over event only if no event occurs during last counting period

Bit 1 **EEVYCRES**: External event Y counter reset
This bit resets the external event Y counter. It is set by software and reset by hardware.
0: No action
1: External event counter Y is reset

Bit 0 **EEVYCE**: External event Y counter enable
This bit enables the external event Y counter.
0: External event Y counter disabled
1: External event Y counter enabled

40.5.56 HRTIM timer x external event selection register (HRTIM_EESELxR) (x = A to F)

Address offset: Block A: 0x174

Address offset: Block B: 0x274

Address offset: Block C: 0x374

Address offset: Block D: 0x474

Address offset: Block E: 0x574

Address offset: Block F: 0x674

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEVJSEL[1:0]		Res.	EEVISEL[1:0]		Res.	EEVHSEL[1:0]		Res.	EEVGSEL[1:0]		Res.	EEVFSEL[1:0]	
		rw	rw		rw	rw		rw	rw		rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EEVESEL[1:0]		Res.	EEVDSEL[1:0]		Res.	EEVCSEL[1:0]		Res.	EEVBSEL[1:0]		Res.	EEVASEL[1:0]	
		rw	rw		rw	rw		rw	rw		rw	rw		rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **EEVJSEL[1:0]**: External event J selection
00: EEVJ is connected to EEV10
01: EEVJ is connected to EEV14
10: EEVJ is connected to (EEV10 AND EEV11)
11: EEVJ is connected to (EEV10 AND EEV14)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 27 Reserved, must be kept at reset value.

Bits 26:25 **EEVISEL[1:0]**: External event I selection
00: EEVI is connected to EEV9
01: EEVI is connected to EEV13
10: EEVI is connected to (EEV9 AND EEV10)
11: EEVI is connected to (EEV9 AND EEV13)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 24 Reserved, must be kept at reset value.

Bits 23:22 **EEVHSEL[1:0]**: External event H selection
00: EEVH is connected to EEV8
01: EEVH is connected to EEV12
10: EEVH is connected to (EEV8 AND EEV9)
11: EEVH is connected to (EEV8 AND EEV12)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 21 Reserved, must be kept at reset value.

Bits 20:19 **EEVGSEL[1:0]**: External event G selection
00: EEVG is connected to EEV7
01: EEVG is connected to EEV11
10: EEVG is connected to (EEV7 AND EEV8)
11: EEVG is connected to (EEV7 AND EEV11)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **EEVFSEL[1:0]**: External event F selection
00: EEVF is connected to EEV6
01: EEVF is connected to EEV16
10: EEVF is connected to (EEV6 AND EEV7)
11: EEVF is connected to (EEV6 AND EEV16)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **EEVESEL[1:0]**: External event E selection
00: EEVE is connected to EEV5
01: EEVE is connected to EEV15
10: EEVE is connected to (EEV5 AND EEV6)
11: EEVE is connected to (EEV5 AND EEV15)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EEVDSEL[1:0]**: External event D selection
00: EEVD is connected to EEV4
01: EEVD is connected to EEV14
10: EEVD is connected to (EEV4 AND EEV5)
11: EEVD is connected to (EEV4 AND EEV14)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **EEVCSEL[1:0]**: External event C selection
00: EEVC is connected to EEV3
01: EEVC is connected to EEV13
10: EEVC is connected to (EEV3 AND EEV4)
11: EEVC is connected to (EEV3 AND EEV13)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EEVBSEL[1:0]**: External event B selection
00: EEVB is connected to EEV2
01: EEVB is connected to EEV12
10: EEVB is connected to (EEV2 AND EEV3)
11: EEVB is connected to (EEV2 AND EEV12)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **EEVASEL[1:0]**: External event A selection
00: EEVA is connected to EEV1
01: EEVA is connected to EEV11
10: EEVA is connected to (EEV1 AND EEV2)
11: EEVA is connected to (EEV1 AND EEV11)
Note: This bitfield must not be modified once the counter is enabled (TxCEN bit set).

40.5.57 HRTIM timer x crossbar override trigger register (HRTIM_COTRGxR) (x = A to F)

Address offset: Block A: 0x178

Address offset: Block B: 0x278

Address offset: Block C: 0x378

Address offset: Block D: 0x478

Address offset: Block E: 0x578

Address offset: Block F: 0x678

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXT_TRG5	EXT_TRG4	EXT_TRG3	EXT_TRG2	EXT_TRG1	EXT_TRG0	EEVJ	EEVI	EEVH	EEVG	EEVF	EEVE	EEVD	EEVC	EEVB	EEVA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT7A8	FLT6A7	FLT5A6	FLT4A5	FLT3A4	FLT2A3	FLT1A2	FLT8	FLT7	FLT6	FLT5	FLT4	FLT3	FLT2	FLT1	SWT
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rs

Bit 31 **EXT_TRG5**: External trigger 5
0: External trigger 5 ignored
1: External trigger 5 enabled

Bit 30 **EXT_TRG4**: External trigger 4
0: External trigger 4 ignored
1: External trigger 4 enabled

Bit 29 **EXT_TRG3**: External trigger 3
0: External trigger 3 ignored
1: External trigger 3 enabled

Bit 28 **EXT_TRG2**: External trigger 2
0: External trigger 2 ignored
1: External trigger 2 enabled

Bit 27 **EXT_TRG1**: External trigger 1
0: External trigger 1 ignored
1: External trigger 1 enabled

Bit 26 **EXT_TRG0**: External trigger 0
0: External trigger 0 ignored
1: External trigger 0 enabled

Bit 25 **EEVJ**: EEVJ trigger
0: EEVJ input trigger ignored
1: EEVJ input trigger enabled

Bit 24 **EEVI**: EEVI trigger
0: EEVI input trigger ignored
1: EEVI input trigger enabled

Bit 23 **EEVH**: EEVH trigger
0: EEVH input trigger ignored
1: EEVH input trigger enabled

Bit 22 **EEVG**: EEVG trigger
0: EEVG input trigger ignored
1: EEVG input trigger enabled

Bit 21 **EEVF**: EEVF trigger
0: EEVF input trigger ignored
1: EEVF input trigger enabled

Bit 20 **EEVE**: EEVE trigger
0: EEVE input trigger ignored
1: EEVE input trigger enabled

Bit 19 **EEVD**: EEVD trigger
0: EEVD input trigger ignored
1: EEVD input trigger enabled

Bit 18 **EEVC**: EEVC trigger
0: EEVC input trigger ignored
1: EEVC input trigger enabled

Bit 17 **EEVB**: EEVB trigger
0: EEVB input trigger ignored
1: EEVB input trigger enabled

Bit 16 **EEVA**: EEVA trigger
0: EEVA input trigger ignored
1: EEVA input trigger enabled

Bit 15 **FLT7A8**: FLT7A8 trigger
0: (Fault 7 AND Fault 8) trigger ignored
1: (Fault 7 AND Fault 8) trigger enabled

Bit 14 **FLT6A7**: FLT6A7 trigger
0: (Fault 6 AND Fault 7) trigger ignored
1: (Fault 6 AND Fault 7) trigger enabled

Bit 13 **FLT5A6**: FLT5A6 trigger
0: (Fault 5 AND Fault 6) trigger ignored
1: (Fault 5 AND Fault 6) trigger enabled

Bit 12 **FLT4A5**: FLT4A5 trigger
0: (Fault 4 AND Fault 5) trigger ignored
1: (Fault 4 AND Fault 5) trigger enabled

Bit 11 **FLT3A4**: FLT2A3 trigger
0: (Fault 3 AND Fault 4) trigger ignored
1: (Fault 3 AND Fault 4) trigger enabled

Bit 10 **FLT2A3**: FLT2A3 trigger
0: (Fault 2 AND Fault 3) trigger ignored
1: (Fault 2 AND Fault 3) trigger enabled

Bit 9 **FLT1A2**: FLT1A2 trigger
0: (Fault 1 AND Fault 2) trigger ignored
1: (Fault 1 AND Fault 2) trigger enabled

Bit 8 **FLT8**: FLT8 trigger
0: Fault 8 input trigger ignored
1: Fault 8 input trigger enabled

Bit 7 **FLT7**: FLT7 trigger
0: Fault 7 input trigger ignored
1: Fault 7 input trigger enabled

Bit 6 **FLT6**: FLT6 trigger
0: Fault 6 input trigger ignored
1: Fault 6 input trigger enabled

Bit 5 **FLT5**: FLT5 trigger
0: Fault 5 input trigger ignored
1: Fault 5 input trigger enabled

Bit 4 **FLT4**: FLT4 trigger
0: Fault 4 input trigger ignored
1: Fault 4 input trigger enabled

Bit 3 **FLT3**: FLT3 trigger
0: Fault 3 input trigger ignored
1: Fault 3 input trigger enabled

Bit 2 **FLT2**: FLT2 trigger
0: Fault 2 input trigger ignored
1: Fault 2 input trigger enabled

Bit 1 **FLT1**: FLT1 trigger
0: Fault 1 input trigger ignored
1: Fault 1 input trigger enabled

Bit 0 **SWT**: Software trigger
0: No action
1: Crossbar override software trigger

40.5.58 HRTIM timer x crossbar override forcing event register (HRTIM_COFExR) (x = A to F)

Address offset: Block A: 0x17C

Address offset: Block B: 0x27C

Address offset: Block C: 0x37C

Address offset: Block D: 0x47C

Address offset: Block E: 0x57C

Address offset: Block F: 0x67C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EEVJFRCB	EEVIFRCB	EEVHFRCB	EEVGFRCB	EEVFFRCB	EEVEFRCB	EEVDFRCB	EEVCFRCB	EEVBFRCB	EEVAFRCB	Res.	COFMB[2:0]		
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EEVJFRCA	EEVIFRCA	EEVHFRCA	EEVGFRCA	EEVFFRCA	EEVEFRCA	EEVDFRCA	EEVCFRCA	EEVBFRCA	EEVAFRCA	Res.	COFMA[2:0]		
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **EEVJFRCB**: External event J forces output B
0: No action
1: Output B is forced to the active level when External Event J is active

Bit 28 **EEVIFRCB**: External event I forces output B
0: No action
1: Output B is forced to the active level when External Event I is active

Bit 27 **EEVHFRCB**: External event H forces output B
0: No action
1: Output B is forced to the active level when External Event H is active

Bit 26 **EEVGFRCB**: External event G forces output B
0: No action
1: Output B is forced to the active level when External Event G is active

Bit 25 **EEVFFRCB**: External event F forces output B
0: No action
1: Output B is forced to the active level when External Event F is active

Bit 24 **EEVEFRCB**: External event E forces output B
0: No action
1: Output B is forced to active level when External Event E is active

Bit 23 **EEVDFRCB**: External event D forces output B
0: No action
1: Output B is forced to the active level when External Event D is active

Bit 22 **EEVCFRCB**: External event C forces output B
0: No action
1: Output B is forced to the active level when External Event C is active

Bit 21 **EEVBFRCB**: External event B forces output B
0: No action
1: Output B is forced to the active level when External Event B is active

Bit 20 **EEVAFRCB**: External event A forces output B
0: No action
1: Output B is forced to active level when External Event A is active

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **COFMB[2:0]**: Crossbar override force mode output B
This bitfield defines how the output B is handled once the crossbar override mode is triggered
000: Delayed 1 fault
Output B is forced to the inactive level after Delay 1 duration defined with DLY1FLTD[8:0] bitfield in HRTIM_CODExR register.
001: Delayed 2 fault
Output B is forced to the inactive level after Delay 2 duration defined with DLY2FLTD[8:0] bitfield in HRTIM_CODExR register
010: Output B forced to inactive level
011: Output B forced to active level
100: Output B forced on events, as defined with EEVAFRCB..EEVJFRCB bits.
Others: reserved

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **EEVJFRCA**: External event J forces output A
0: No action
1: Output A is forced to the active level when External Event J is active

Bit 12 **EEVIFRCA**: External event I forces output A
0: No action
1: Output A is forced to the active level when External Event I is active

Bit 11 **EEVHFRCA**: External event H forces output A
0: No action
1: Output A is forced to the active level when External Event H is active

Bit 10 **EEVGFRCA**: External event G forces output A
0: No action
1: Output A is forced to the active level when External Event G is active

Bit 9 **EEVFFRCA**: External event F forces output A
0: No action
1: Output A is forced to the active level when External Event F is active

Bit 8 **EEVEFRCA**: External event E forces output A
0: No action
1: Output A is forced to active level when External Event E is active

Bit 7 **EEVDFRCA**: External event D forces output A
0: No action
1: Output A is forced to the active level when External Event D is active

Bit 6 **EEVCFRCA**: External event C forces output A
0: No action
1: Output A is forced to the active level when External Event C is active

Bit 5 **EEVBFRCA**: External event B forces output A
0: No action
1: Output A is forced to the active level when External Event B is active

Bit 4 **EEVAFRCA**: External event A forces output A
0: No action
1: Output A is forced to active level when External Event A is active

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **COFMA[2:0]**: Crossbar override force mode output A
This bitfield defines how the output A is handled once the crossbar override mode is triggered
000: Delayed 1 fault
Output A is forced to the inactive level after Delay 1 duration defined with DLY1FLTD[8:0] bitfield in HRTIM_CODExR register.
001: Delayed 2 fault
Output A is forced to the inactive level after Delay 2 duration defined with DLY2FLTD[8:0] bitfield in HRTIM_CODExR register
010: Output A forced to inactive level
011: Output A forced to active level
100: Output A forced on events, as defined with EEVAFRCA..EEVJFRCA bits.
Others: reserved

40.5.59 HRTIM timer x crossbar override delayed fault event register (HRTIM_CODExR) (x = A to F)

Address offset: Block A: 0x180

Address offset: Block B: 0x280

Address offset: Block C: 0x380

Address offset: Block D: 0x480

Address offset: Block E: 0x580

Address offset: Block F: 0x680

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLY2FLTD[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLY1FLTD[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **DLY2FLTD[8:0]**: Delayed 2 Fault duration

This register holds the value of the delay applied before entering in Fault mode, for Delayed 1 fault mode.

t_{Delay2} = DLY2FLTD[8:0] x t_{DTG}

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **DLY1FLTD[8:0]**: Delayed 1 Fault duration

This register holds the value of the delay applied before entering in Fault mode, for Delayed 1 fault mode.

t_{Delay1} = DLY1FLTD[8:0] x t_{DTG}

40.5.60 HRTIM control register 1 (HRTIM_CR1)

Address offset: 0x700

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]		
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RTFUDIS	RTEUDIS	RTDUDIS	RTCUDIS	RTBUDIS	RTAUDIS	RMUDIS	Res.	TFUDIS	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:25 **AD4USRC[2:0]**: ADC trigger 4 update source
Refer to AD1USRC[2:0] description.

Bits 24:22 **AD3USRC[2:0]**: ADC trigger 3 update source
Refer to AD1USRC[2:0] description.

Bits 21:19 **AD2USRC[2:0]**: ADC trigger 2 update source
Refer to AD1USRC[2:0] description.

Bits 18:16 **AD1USRC[2:0]**: ADC trigger 1 update source
These bits define the source which triggers the update of the HRTIM_ADC1R register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, in HRTIM_MCR or HRTIM_TIMxCR.
000: Master timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110: Timer F
111: Reserved

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RTFUDIS**: Remote Timer F update disable
Refer to RTAUDIS description.

Bit 13 **RTEUDIS**: Remote Timer E update disable
Refer to RTAUDIS description

Bit 12 **RTDUDIS**: Remote Timer D update disable
Refer to RTAUDIS description.

Bit 11 **RTCUDIS**: Remote Timer C update disable
Refer to RTAUDIS description.

Bit 10 **RTBUDIS**: Remote Timer B update disable
Refer to RTAUDIS description.

Bit 9 **RTAUDIS**: Remote timer A update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily in another HRTIMER's timer A.

0: Update enabled. The update occurs upon generation of the selected source.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 8 **RMUDIS**: Remote master update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily in another HRTIMER's master timer

0: Update enabled.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TFUDIS**: Timer F update disable

Refer to TAUDIS description.

Bit 5 **TEUDIS**: Timer E update disable

Refer to TAUDIS description

Bit 4 **TDUDIS**: Timer D update disable

Refer to TAUDIS description.

Bit 3 **TCUDIS**: Timer C update disable

Refer to TAUDIS description.

Bit 2 **TBUDIS**: Timer B update disable

Refer to TAUDIS description.

Bit 1 **TAUDIS**: Timer A update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily on timer A.

0: Update enabled. The update occurs upon generation of the selected source.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

Bit 0 **MUDIS**: Master update disable

This bit is set and cleared by software to enable/disable an update event generation temporarily.

0: Update enabled.

1: Update disabled. The updates are temporarily disabled to allow the software to write multiple registers that have to be simultaneously taken into account.

40.5.61 HRTIM control register 2 (HRTIM_CR2)

Address offset: 0x704

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	DBSS[2:0]			Res.	Res.	SWPF	SWPE	SWPD	SWPC	SWPB	SWPA
					rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TFRST	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	TFSWU	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 DBSS[2:0]: DMA burst source selection

This bitfield defines the DMA source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done at the HRTIM_BDMADR register address).

000:Reserved
001:Master timer
010:Timer A
011:Timer B
100:Timer C
101:Timer D
110:Timer E
111:Timer F

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SWPF**: Swap timer F outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 20 **SWPE**: Swap timer E outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 19 **SWPD**: Swap timer D outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 18 **SWPC**: Swap timer C outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 17 **SWPB**: Swap timer B outputs

Refer to SWPA description.

Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 16 **SWPA**: Swap timer A outputs
This bit allows to swap the timer A outputs.
0: HRTIM_SETA1R and HRTIM_RSTA1R are coding for the output A1, HRTIM_SETA2R and HRTIM_RSTA2R are coding for the output A2
1: HRTIM_SETA1R and HRTIM_RSTA1R are coding for the output A2, HRTIM_SETA2R and HRTIM_RSTA2R are coding for the output A1
Note: This bit is not significant when the Push-pull mode is enabled (PSHPLL = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TFRST**: Timer F counter software reset
Refer to TARST description.

Bit 13 **TERST**: Timer E counter software reset
Refer to TARST description.

Bit 12 **TDRST**: Timer D counter software reset
Refer to TARST description.

Bit 11 **TCRST**: Timer C counter software reset
Refer to TARST description.

Bit 10 **TBRST**: Timer B counter software reset
Refer to TARST description.

Bit 9 **TARST**: Timer A counter software reset
Setting this bit resets the timer A counter.
The bit is automatically reset by hardware.

Bit 8 **MRST**: Master counter software reset
Setting this bit resets the master timer counter.
The bit is automatically reset by hardware.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TFSWU**: Timer F software update
Refer to TASWU description.

Bit 5 **TESWU**: Timer E software update
Refer to TASWU description.

Bit 4 **TDSWU**: Timer D software update
Refer to TASWU description.

Bit 3 **TCSWU**: Timer C software update
Refer to TASWU description.

Bit 2 **TBSWU**: Timer B software update
Refer to TASWU description.

Bit 1 **TASWU**: Timer A software update
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register and any pending update request is canceled.

Bit 0 **MSWU**: Master timer software update
This bit is set by software and automatically reset by hardware. It forces an immediate transfer from the preload to the active register in the master timer and any pending update request is canceled.

40.5.62 HRTIM interrupt status register (HRTIM_ISR)

Address offset: 0x708

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	DLLRDY
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8	FLT7	FLT6	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
							r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPER**: Burst mode period interrupt flag
This bit is set by hardware when a single-shot burst mode operation is completed or at the end of a burst mode period in continuous mode. It is cleared by software writing it at 1.
0: No burst mode period interrupt occurred
1: Burst mode period interrupt occurred

Bit 16 **DLLRDY**: DLL Ready Interrupt Flag
This bit is set by hardware when the DLL calibration is completed. It is cleared by software writing it at 1.
0: No DLL calibration ready interrupt occurred
1: DLL calibration ready interrupt occurred

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **FLT8**: Fault 8 interrupt flag
Refer to FLT1 description.

Bit 7 **FLT7**: Fault 7 interrupt flag
Refer to FLT1 description.

Bit 6 **FLT6**: Fault 6 interrupt flag
Refer to FLT1 description.

Bit 5 **SYSFLT**: System fault interrupt flag
Refer to FLT1 description.

Bit 4 **FLT5**: Fault 5 interrupt flag
Refer to FLT1 description.

Bit 3 **FLT4**: Fault 4 interrupt flag
Refer to FLT1 description.

Bit 2 **FLT3**: Fault 3 interrupt flag
Refer to FLT1 description.

Bit 1 **FLT2**: Fault 2 interrupt flag
Refer to FLT1 description.

Bit 0 **FLT1**: Fault 1 interrupt flag
This bit is set by hardware when fault 1 event occurs. It is cleared by software writing it at 1.
0: No fault 1 interrupt occurred
1: Fault 1 interrupt occurred

40.5.63 HRTIM interrupt clear register (HRTIM_ICR)

Address offset: 0x70C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	DLLRDYC
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8C	FLT7C	FLT6C	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
							w	w	w	w	w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPERC**: Burst mode period flag clear
Writing 1 to this bit clears the BMPER flag in HRTIM_ISR register.

Bit 16 **DLLRDYC**: DLL Ready Interrupt flag Clear
Writing 1 to this bit clears the DLLRDY flag in HRTIM_ISR register.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **FLT8C**: Fault 8 interrupt flag clear
Writing 1 to this bit clears the FLT6 flag in HRTIM_ISR register.

Bit 7 **FLT7C**: Fault 7 interrupt flag clear
Writing 1 to this bit clears the FLT6 flag in HRTIM_ISR register.

Bit 6 **FLT6C**: Fault 6 interrupt flag clear
Writing 1 to this bit clears the FLT6 flag in HRTIM_ISR register.

Bit 5 **SYSFLTC**: System fault interrupt flag clear
Writing 1 to this bit clears the SYSFLT flag in HRTIM_ISR register.

Bit 4 **FLT5C**: Fault 5 interrupt flag clear
Writing 1 to this bit clears the FLT5 flag in HRTIM_ISR register.

Bit 3 **FLT4C**: Fault 4 interrupt flag clear
Writing 1 to this bit clears the FLT4 flag in HRTIM_ISR register.

Bit 2 **FLT3C**: Fault 3 interrupt flag clear
Writing 1 to this bit clears the FLT3 flag in HRTIM_ISR register.

Bit 1 **FLT2C**: Fault 2 interrupt flag clear
Writing 1 to this bit clears the FLT2 flag in HRTIM_ISR register.

Bit 0 **FLT1C**: Fault 1 interrupt flag clear
Writing 1 to this bit clears the FLT1 flag in HRTIM_ISR register.

40.5.64 HRTIM interrupt enable register (HRTIM_IER)

Address offset: 0x710

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	DLLRDYIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8IE	FLT7IE	FLT6IE	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **BMPERIE**: Burst mode period interrupt enable
This bit is set and cleared by software to enable/disable the burst mode period interrupt.
0: Burst mode period interrupt disabled
1: Burst mode period interrupt enabled

Bit 16 **DLLRDYIE**: DLL Ready Interrupt Enable
This bit is set and cleared by software to enable/disable the DLL ready interrupt.
0: DLL ready interrupt disabled
1: DLL ready interrupt enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **FLT8IE**: Fault 8 interrupt enable
Refer to FLT1IE description.

Bit 7 **FLT7IE**: Fault 7 interrupt enable
Refer to FLT1IE description.

Bit 6 **FLT6IE**: Fault 6 interrupt enable
Refer to FLT1IE description.

Bit 5 **SYSFLTIE**: System fault interrupt enable
Refer to FLT1IE description.

Bit 4 **FLT5IE**: Fault 5 interrupt enable
Refer to FLT1IE description.

Bit 3 **FLT4IE**: Fault 4 interrupt enable
Refer to FLT1IE description.

Bit 2 **FLT3IE**: Fault 3 interrupt enable
Refer to FLT1IE description.

Bit 1 **FLT2IE**: Fault 2 interrupt enable
Refer to FLT1IE description.

Bit 0 **FLT1IE**: Fault 1 interrupt enable
This bit is set and cleared by software to enable/disable the fault 1 interrupt.
0: Fault 1 interrupt disabled
1: Fault 1 interrupt enabled

40.5.65 HRTIM output enable register (HRTIM_OENR)

Address offset: 0x714

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2OEN	TF1OEN	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
				rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2OEN**: Timer F output 2 enable
Refer to TA1OEN description.

Bit 10 **TF1OEN**: Timer F output 1 enable
Refer to TA1OEN description

Bit 9 **TE2OEN**: Timer E output 2 enable
Refer to TA1OEN description.

Bit 8 **TE1OEN**: Timer E output 1 enable
Refer to TA1OEN description.

Bit 7 **TD2OEN**: Timer D output 2 enable
Refer to TA1OEN description.

Bit 6 **TD1OEN**: Timer D output 1 enable
Refer to TA1OEN description.

Bit 5 **TC2OEN**: Timer C output 2 enable
Refer to TA1OEN description.

Bit 4 **TC1OEN**: Timer C output 1 enable
Refer to TA1OEN description.

Bit 3 **TB2OEN**: Timer B output 2 enable
Refer to TA1OEN description.

Bit 2 **TB1OEN**: Timer B output 1 enable
Refer to TA1OEN description.

Bit 1 **TA2OEN**: Timer A output 2 enable
Refer to TA1OEN description.

Bit 0 **TA1OEN**: Timer A output 1 enable
Setting this bit enables the timer A output 1. Writing “0” has no effect.
Reading the bit returns the output enable/disable status.
This bit is cleared asynchronously by hardware as soon as the timer-related fault input(s) is (are) active.
0: output A1 disabled. The output is either in fault or Idle state.
1: output A1 enabled

Note: The disable status corresponds to both idle and fault states. The output disable status is given by TA1ODS bit in the HRTIM_ODSR register.

40.5.66 HRTIM output disable register (HRTIM_ODISR)

Address offset: 0x718

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2ODIS	TF1ODIS	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
				w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2ODIS**: Timer F output 2 disable
Refer to TA1ODIS description.

Bit 10 **TF1ODIS**: Timer F output 1 disable
Refer to TA1ODIS description

Bit 9 **TE2ODIS**: Timer E output 2 disable
Refer to TA1ODIS description.

Bit 8 **TE1ODIS**: Timer E output 1 disable
Refer to TA1ODIS description.

Bit 7 **TD2ODIS**: Timer D output 2 disable
Refer to TA1ODIS description.

Bit 6 **TD1ODIS**: Timer D output 1 disable
Refer to TA1ODIS description.

Bit 5 **TC2ODIS**: Timer C output 2 disable
Refer to TA1ODIS description.

Bit 4 **TC1ODIS**: Timer C output 1 disable
Refer to TA1ODIS description.

Bit 3 **TB2ODIS**: Timer B output 2 disable
Refer to TA1ODIS description.

Bit 2 **TB1ODIS**: Timer B output 1 disable
Refer to TA1ODIS description.

Bit 1 **TA2ODIS**: Timer A output 2 disable
Refer to TA1ODIS description.

Bit 0 **TA1ODIS**: Timer A output 1 disable
Setting this bit disables the timer A output 1. The output enters the idle state, either from the run state or from the fault state.
Writing “0” has no effect.

40.5.67 HRTIM output disable status register (HRTIM_ODSR)

Address offset: 0x71C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TF2ODS	TF1ODS	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TF2ODS**: Timer F output 2 disable status
Refer to TA1ODS description.

Bit 10 **TF1ODS**: Timer F output 1 disable status
Refer to TA1ODS description.

Bit 9 **TE2ODS**: Timer E output 2 disable status
Refer to TA1ODS description.

Bit 8 **TE1ODS**: Timer E output 1 disable status
Refer to TA1ODS description.

Bit 7 **TD2ODS**: Timer D output 2 disable status
Refer to TA1ODS description.

Bit 6 **TD1ODS**: Timer D output 1 disable status
Refer to TA1ODS description.

Bit 5 **TC2ODS**: Timer C output 2 disable status
Refer to TA1ODS description.

Bit 4 **TC1ODS**: Timer C output 1 disable status
Refer to TA1ODS description.

Bit 3 **TB2ODS**: Timer B output 2 disable status
Refer to TA1ODS description.

Bit 2 **TB1ODS**: Timer B output 1 disable status
Refer to TA1ODS description.

Bit 1 **TA2ODS**: Timer A output 2 disable status
Refer to TA1ODS description.

Bit 0 **TA1ODS**: Timer A output 1 disable status
Reading the bit returns the output disable status. It is not significant when the output is active (Tx1OEN or Tx2OEN = 1).
0: Output A1 disabled, in Idle state.
1: Output A1 disabled, in fault state.

40.5.68 HRTIM burst mode control register (HRTIM_BMCR)

Address offset: 0x720

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFBM	TEBM	TDBM	TCBM	TBBM	TABM	MTBM
rc_w0									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BMSTAT**: Burst mode status

This bit gives the current operating state.

0: Normal operation

1: Burst operation on-going. Writing this bit to 0 causes a burst mode early termination.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **TFBM**: Timer F burst mode

Refer to TABM description.

Bit 21 **TEBM**: Timer E burst mode

Refer to TABM description.

Bit 20 **TDBM**: Timer D burst mode

Refer to TABM description.

Bit 19 **TCBM**: Timer C burst mode

Refer to TABM description.

Bit 18 **TBBM**: Timer B burst mode

Refer to TABM description.

Bit 17 **TABM**: Timer A burst mode

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: TA counter clock is maintained and the timer operates normally

1: TA counter clock is stopped and the counter is reset

Note: This bit must not be set when the balanced idle mode is active (DLYPRT[2:0] = 0x11).

Bit 16 **MTBM**: Master timer burst mode

This bit defines how the timer behaves during a burst mode operation. This bitfield cannot be changed while the burst mode is enabled.

0: Master Timer counter clock is maintained and the timer operates normally

1: Master Timer counter clock is stopped and the counter is reset

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **BMPREN**: Burst mode preload enable

This bit enables the registers preload mechanism and defines whether a write access into a preloadable register (HRTIM_BMCMPR, HRTIM_BMPER) is done into the active or the preload register.

0: Preload disabled: the write access is directly done into active registers

1: Preload enabled: the write access is done into preload registers

Bits 9:6 **BMPRSC[3:0]**: Burst mode prescaler

Defines the prescaling ratio of the f_{HRTIM} clock for the burst mode controller. This bitfield cannot be changed while the burst mode is enabled.

0000: Clock not divided
0001: Division by 2
0010: Division by 4
0011: Division by 8
0100: Division by 16
0101: Division by 32
0110: Division by 64
0111: Division by 128
1000: Division by 256
1001: Division by 512
1010: Division by 1024
1011: Division by 2048
1100: Division by 4096
1101: Division by 8192
1110: Division by 16384
1111: Division by 32768

Bits 5:2 **BMCLK[3:0]**: Burst mode clock source

This bitfield defines the clock source for the burst mode counter. It cannot be changed while the burst mode is enabled (refer to *Table 413* for on-chip events 1..4 connections details).

0000: Master timer counter reset/roll-over
0001: Timer A counter reset/roll-over
0010: Timer B counter reset/roll-over
0011: Timer C counter reset/roll-over
0100: Timer D counter reset/roll-over
0101: Timer E counter reset/roll-over
0110: On-chip event 1 (hrtim_bm_ck1), acting as a burst mode counter clock
0111: On-chip event 2 (hrtim_bm_ck2) acting as a burst mode counter clock
1000: On-chip event 3 (hrtim_bm_ck3) acting as a burst mode counter clock
1001: On-chip event 4 (hrtim_bm_ck4) acting as a burst mode counter clock
1010: Prescaled f_{HRTIM} clock (as per BMPRSC[3:0] setting)
1011: Timer F counter reset/roll-over
Others: Reserved

Bit 1 **BMOM**: Burst mode operating mode

This bit defines if the burst mode is entered once or if it is continuously operating.

0: Single-shot mode
1: Continuous operation

Bit 0 **BME**: Burst mode enable

This bit starts the burst mode controller which becomes ready to receive the start trigger. Writing this bit to 0 causes a burst mode early termination.

0: Burst mode disabled
1: Burst mode enabled

40.5.69 HRTIM burst mode trigger register (HRTIM_BMTRGR)

Address offset: 0x724

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OCHPEV	EEVH	EEVG	TDEEVH	TAEEVG	TECMP2	TECMP1	TEREP	TFCMP1	TDCMP2	TFREP	TDREP	TDRST	TFRST	TCCMP1	TCREP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OCHPEV**: On-chip event

A rising edge on the hrtim_bm_trg input (connected to general purpose timer TRGO output) triggers a burst mode entry (see *Table 410* for details).

Bit 30 **EEVH**: External event H (TIMD filters applied)

The external event H conditioned by TIMD filters is starting the burst mode operation.

Bit 29 **EEVG**: External event G (TIMA filters applied)

The external event G conditioned by TIMA filters is starting the burst mode operation.

Bit 28 **TDEEVH**: Timer D period following external event H

The timer D period following an external event H (conditioned by TIMD filters) is starting the burst mode operation.

Bit 27 **TAEEVG**: Timer A period following external event G

The timer A period following an external event G (conditioned by TIMA filters) is starting the burst mode operation.

Bit 26 **TECMP2**: Timer E compare 2 event

Refer to TACMP1 description.

Bit 25 **TECMP1**: Timer E compare 1 event

Refer to TACMP1 description.

Bit 24 **TEREP**: Timer E repetition

Refer to TAREP description.

Bit 23 **TFCMP1**: Timer F compare 1 event

Refer to TACMP1 description.

Bit 22 **TDCMP2**: Timer D compare 2 event

Refer to TACMP1 description.

Bit 21 **TFREP**: Timer F repetition

Refer to TAREP description.

Bit 20 **TDREP**: Timer D repetition

Refer to TAREP description.

Bit 19 **TDRST**: Timer D reset or roll-over

Refer to TARST description.

Bit 18 **TFRST**: Timer F reset

Refer to TARST description.

Bit 17 **TCCMP1**: Timer C compare 1 event
Refer to TACMP1 description.

Bit 16 **TCREP**: Timer C repetition
Refer to TAREP description.

Bit 15 **TCRST**: Timer C reset or roll-over
Refer to TARST description.

Bit 14 **TBCMP2**: Timer B compare 2 event
Refer to TACMP1 description.

Bit 13 **TBCMP1**: Timer B compare 1 event
Refer to TACMP1 description.

Bit 12 **TBREP**: Timer B repetition
Refer to TAREP description.

Bit 11 **TBRST**: Timer B reset or roll-over
Refer to TARST description.

Bit 10 **TACMP2**: Timer A compare 2 event
Refer to TACMP1 description.

Bit 9 **TACMP1**: Timer A compare 1 event
The timer A compare 1 event is starting the burst mode operation.

Bit 8 **TAREP**: Timer A repetition
The Timer A repetition event is starting the burst mode operation.

Bit 7 **TARST**: Timer A reset or roll-over
The Timer A reset or roll-over event is starting the burst mode operation.

Bit 6 **MSTCMP4**: Master compare 4
Refer to MSTCMP1 description.

Bit 5 **MSTCMP3**: Master compare 3
Refer to MSTCMP1 description.

Bit 4 **MSTCMP2**: Master compare 2
Refer to MSTCMP1 description.

Bit 3 **MSTCMP1**: Master compare 1
The master timer compare 1 event is starting the burst mode operation.

Bit 2 **MSTREP**: Master repetition
The master timer repetition event is starting the burst mode operation.

Bit 1 **MSTRST**: Master reset or roll-over
The master timer reset and roll-over event is starting the burst mode operation.

Bit 0 **SW**: Software start
This bit is set by software and automatically reset by hardware.
When set, It starts the burst mode operation immediately.
This bit is not active if the burst mode is not enabled (BME bit is reset).

40.5.70 HRTIM burst mode compare register (HRTIM_BMCMPR)

Address offset: 0x728

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMCMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BMCMP[15:0]**: Burst mode compare value

Defines the number of periods during which the selected timers are in idle state.

This register holds either the content of the preload register or the content of the active register if the preload is disabled.

Note: BMCMP[15:0] cannot be set to 0x0000 when using the f_{HRTIM} clock without a prescaler as the burst mode clock source (BMCLK[3:0] = 1010 and BMPRESC[3:0] = 0000).

40.5.71 HRTIM burst mode period register (HRTIM_BMPER)

Address offset: 0x72C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BMPER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BMPER[15:0]**: Burst mode period

Defines the burst mode repetition period.

This register holds either the content of the preload register or the content of the active register if preload is disabled.

Note: The BMPER[15:0] must not be null when the burst mode is enabled.

40.5.72 HRTIM timer external event control register 1 (HRTIM_EECR1)

Address offset: 0x730

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1]
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE3SNS[0]	EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **EE5FAST**: External event 5 fast mode
Refer to EE1FAST description.

Bits 28:27 **EE5SNS[1:0]**: External event 5 sensitivity
Refer to EE1SNS[1:0] description.

Bit 26 **EE5POL**: External event 5 polarity
Refer to EE1POL description.

Bits 25:24 **EE5SRC[1:0]**: External event 5 source
This bitfield is combined with EE5SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 23 **EE4FAST**: External event 4 fast mode
Refer to EE1FAST description.

Bits 22:21 **EE4SNS[1:0]**: External event 4 sensitivity
Refer to EE1SNS[1:0] description.

Bit 20 **EE4POL**: External event 4 polarity
Refer to EE1POL description.

Bits 19:18 **EE4SRC[1:0]**: External event 4 source
This bitfield is combined with EE4SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 17 **EE3FAST**: External event 3 fast mode
Refer to EE1FAST description.

Bits 16:15 **EE3SNS[1:0]**: External event 3 sensitivity
Refer to EE1SNS[1:0] description.

Bit 14 **EE3POL**: External event 3 polarity
Refer to EE1POL description.

Bits 13:12 **EE3SRC[1:0]**: External event 3 source
This bitfield is combined with EE3SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 11 **EE2FAST**: External event 2 fast mode
Refer to EE1FAST description.

Bits 10:9 **EE2SNS[1:0]**: External event 2 sensitivity
Refer to EE1SNS[1:0] description.

Bit 8 **EE2POL**: External event 2 polarity
Refer to EE1POL description.

Bits 7:6 **EE2SRC[1:0]**: External event 2 source
This bitfield is combined with EE2SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 5 **EE1FAST**: External event 1 fast mode
0: External event 1 is re-synchronized by the HRTIM logic before acting on outputs, which adds a f_{HRTIM} clock-related latency
1: External event 1 is acting asynchronously on outputs (low latency mode)
Note: This bit must not be modified once the counter in which the event is used is enabled (TxCEN bit set).

Bits 4:3 **EE1SNS[1:0]**: External event 1 sensitivity
00: On active level defined by EE1POL bit
01: Rising edge, whatever EE1POL bit value
10: Falling edge, whatever EE1POL bit value
11: Both edges, whatever EE1POL bit value

Bit 2 **EE1POL**: External event 1 polarity
This bit is only significant if EE1SNS[1:0] = 00.
0: External event is active high
1: External event is active low
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

Bits 1:0 **EE1SRC[1:0]**: External event 1 source
This bitfield is combined with EE1SRC[2] bit. This bitfield selects the External event 1 source. See *Table 411* for details.
000:hrtim_eev1_1
001:hrtim_eev1_2
010:hrtim_eev1_3
011:hrtim_eev1_4
100:hrtim_eev1_5
101:hrtim_eev1_6
110:hrtim_eev1_7
111:hrtim_eev1_8
Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE1FAST bit.

40.5.73 HRTIM timer external event control register 2 (HRTIM_EECR2)

Address offset: 0x734

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8SNS[0]	EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:27 **EE10SNS[1:0]**: External event 10 sensitivity
Refer to EE1SNS[1:0] description.

Bit 26 **EE10POL**: External event 10 polarity
Refer to EE1POL description.

Bits 25:24 **EE10SRC[1:0]**: External event 10 source
This bitfield is combined with EE10SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EE9SNS[1:0]**: External event 9 sensitivity
Refer to EE1SNS[1:0] description.

Bit 20 **EE9POL**: External event 9 polarity
Refer to EE1POL description.

Bits 19:18 **EE9SRC[1:0]**: External event 9 source
This bitfield is combined with EE9SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:15 **EE8SNS[1:0]**: External event 8 sensitivity
Refer to EE1SNS[1:0] description.

Bit 14 **EE8POL**: External event 8 polarity
Refer to EE1POL description.

Bits 13:12 **EE8SRC[1:0]**: External event 8 source
This bitfield is combined with EE8SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EE7SNS[1:0]**: External event 7 sensitivity
Refer to EE1SNS[1:0] description.

Bit 8 **EE7POL**: External event 7 polarity
Refer to EE1POL description.

Bits 7:6 **EE7SRC[1:0]**: External event 7 source
This bitfield is combined with EE7SRC[2] bit. Refer to EE1SRC[1:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EE6SNS[1:0]**: External event 6 sensitivity
Refer to EE1SNS[1:0] description.

Bit 2 **EE6POL**: External event 6 polarity
Refer to EE1POL description.

Bits 1:0 **EE6SRC[1:0]**: External event 6 source
This bitfield is combined with EE6SRC[2] bit. Refer to EE1SRC[1:0] description.

40.5.74 HRTIM timer external event control register 3 (HRTIM_EECR3)

Address offset: 0x738

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.
rw	rw			rw	rw	rw	rw			rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[3:0]			
rw	rw	rw	rw			rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 **EEVSD[1:0]**: External event sampling clock division
This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the external event signal sampling clock (f_{EEVS}) used by the digital filters.
00: $f_{EEVS}=f_{HRTIM}$
01: $f_{EEVS}=f_{HRTIM}$ / 2
10: $f_{EEVS}=f_{HRTIM}$ / 4
11: $f_{EEVS}=f_{HRTIM}$ / 8

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **EE10F[3:0]**: External event 10 filter
Refer to EE6F[3:0] description.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:18 **EE9F[3:0]**: External event 9 filter
Refer to EE6F[3:0] description.

Bits 17:16 Reserved, must be kept at reset value.

Bits 15:12 **EE8F[3:0]**: External event 8 filter
Refer to EE6F[3:0] description.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **EE7F[3:0]**: External event 7 filter
Refer to EE6F[3:0] description.

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **EE6F[3:0]**: External event 6 filter
This bitfield defines the frequency used to sample external event 6 input and the length of the digital filter applied to EEV6. The digital filter is made of a counter in which N valid samples are needed to validate a transition on the output.
0000: Filter disabled
0001: $f_{SAMPLING} = f_{HRTIM}$, N=2
0010: $f_{SAMPLING} = f_{HRTIM}$, N=4
0011: $f_{SAMPLING} = f_{HRTIM}$, N=8
0100: $f_{SAMPLING} = f_{EEVS}/2$, N=6
0101: $f_{SAMPLING} = f_{EEVS}/2$, N=8
0110: $f_{SAMPLING} = f_{EEVS}/4$, N=6
0111: $f_{SAMPLING} = f_{EEVS}/4$, N=8
1000: $f_{SAMPLING} = f_{EEVS}/8$, N=6
1001: $f_{SAMPLING} = f_{EEVS}/8$, N=8
1010: $f_{SAMPLING} = f_{EEVS}/16$, N=5
1011: $f_{SAMPLING} = f_{EEVS}/16$, N=6
1100: $f_{SAMPLING} = f_{EEVS}/16$, N=8
1101: $f_{SAMPLING} = f_{EEVS}/32$, N=5
1110: $f_{SAMPLING} = f_{EEVS}/32$, N=6
1111: $f_{SAMPLING} = f_{EEVS}/32$, N=8

40.5.75 HRTIM ADC trigger 1 register (HRTIM_ADC1R)

Address offset: 0x73C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC1TEPER	ADC1TEC4	ADC1TEC3	ADC1TFRST	ADC1TDPER	ADC1TDC4	ADC1TDC3	ADC1TFPER	ADC1TCPER	ADC1TCC4	ADC1TCC3	ADC1TFC4	ADC1TBRST	ADC1TBPER	ADC1TBC4	ADC1TBC3
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC1TFC3	ADC1TARST	ADC1TAPER	ADC1TAC4	ADC1TAC3	ADC1TFC2	ADC1EEV5	ADC1EEV4	ADC1EEV3	ADC1EEV2	ADC1EEV1	ADC1MPER	ADC1MC4	ADC1MC3	ADC1MC2	ADC1MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC1TEPER**: ADC trigger 1 on timer E period
Refer to ADC1TAPER description.

Bit 30 **ADC1TEC4**: ADC trigger 1 on timer E compare 4
Refer to ADC1TFC2 description.

Bit 29 **ADC1TEC3**: ADC trigger 1 on timer E compare 3
Refer to ADC1TFC2 description.

Bit 28 **ADC1TFRST**: ADC trigger 1 on timer F reset and counter roll-over(1)
Refer to ADC1TARST description.

Bit 27 **ADC1TDPER**: ADC trigger 1 on timer D period
Refer to ADC1TAPER description.

Bit 26 **ADC1TDC4**: ADC trigger 1 on timer D compare 4
Refer to ADC1TFC2 description.

Bit 25 **ADC1TDC3**: ADC trigger 1 on timer D compare 3
Refer to ADC1TFC2 description.

Bit 24 **ADC1TFPER**: ADC trigger 1 on timer F period
Refer to ADC1TAPER description.

Bit 23 **ADC1TCPER**: ADC trigger 1 on timer C period
Refer to ADC1TAPER description.

Bit 22 **ADC1TCC4**: ADC trigger 1 on timer C compare 4
Refer to ADC1TFC2 description.

Bit 21 **ADC1TCC3**: ADC trigger 1 on timer C compare 3
Refer to ADC1TFC2 description.

Bit 20 **ADC1TFC4**: ADC trigger 1 on timer F compare 4
Refer to ADC1TAC2 description.

Bit 19 **ADC1TBRST**: ADC trigger 1 on timer B reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer B reset and roll-over event, on ADC trigger 1 output.

Bit 18 **ADC1TBPER**: ADC trigger 1 on timer B period
Refer to ADC1TAPER description.

Bit 17 **ADC1TBC4**: ADC trigger 1 on timer B compare 4
Refer to ADC1TFC2 description.

Bit 16 **ADC1TBC3**: ADC trigger 1 on timer B compare 3
Refer to ADC1TFC2 description.

Bit 15 **ADC1TFC3**: ADC trigger 1 on timer F compare 3
Refer to ADC1TFC2 description.

Bit 14 **ADC1TARST**: ADC trigger 1 on timer A reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer A reset and roll-over event, on ADC trigger 1 output.

Bit 13 **ADC1TAPER**: ADC trigger 1 on timer A period
This bit enables the generation of an ADC trigger upon timer A period event, on ADC trigger 1 output.

Bit 12 **ADC1TAC4**: ADC trigger 1 on timer A compare 4
Refer to ADC1TFC2 description.

Bit 11 **ADC1TAC3**: ADC trigger 1 on timer A compare 3
Refer to ADC1TFC2 description.

Bit 10 **ADC1TFC2**: ADC trigger 1 on timer F compare 2
This bit enables the generation of an ADC trigger upon timer F compare 2 event, on ADC trigger 1 output.

Bit 9 **ADC1EEV5**: ADC trigger 1 on external event 5(1)
Refer to ADC1EEV1 description.

Bit 8 **ADC1EEV4**: ADC trigger 1 on external event 4(1)
Refer to ADC1EEV1 description.

Bit 7 **ADC1EEV3**: ADC trigger 1 on external event 3(1)
Refer to ADC1EEV1 description.

Bit 6 **ADC1EEV2**: ADC trigger 1 on external event 2(1)
Refer to ADC1EEV1 description.

Bit 5 **ADC1EEV1**: ADC trigger 1 on external event 1(1)
This bit enables the generation of an ADC trigger upon external event 1, on ADC trigger 1 output.

Bit 4 **ADC1MPER**: ADC trigger 1 on master period
This bit enables the generation of an ADC trigger upon master timer period event, on ADC trigger 1 output.

Bit 3 **ADC1MC4**: ADC trigger 1 on master compare 4
Refer to ADC1MC1 description.

Bit 2 **ADC1MC3**: ADC trigger 1 on master compare 3
Refer to ADC1MC1 description.

Bit 1 **ADC1MC2**: ADC trigger 1 on master compare 2
Refer to ADC1MC1 description.

Bit 0 **ADC1MC1**: ADC trigger 1 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 1 output.

1. These triggers are differing from HRTIM_ADC2R/HRTIM_ADC4R to HRTIM_ADC1R/HRTIM_ADC3R.

40.5.76 HRTIM ADC trigger 2 register (HRTIM_ADC2R)

Address offset: 0x740

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2TERST	ADC2TEC4	ADC2TEC3	ADC2TEC2	ADC2TDRST	ADC2TDPER	ADC2TDC4	ADC2TFPER	ADC2TDC2	ADC2TCRST	ADC2TCPER	ADC2TCC4	ADC2TFC4	ADC2TCC2	ADC2TBPER	ADC2TBC4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC2TFC3	ADC2TBC2	ADC2TAPER	ADC2TAC4	ADC2TFC2	ADC2TAC2	ADC2EEV10	ADC2EEV9	ADC2EEV8	ADC2EEV7	ADC2EEV6	ADC2MPER	ADC2MC4	ADC2MC3	ADC2MC2	ADC2MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC2TERST**: ADC trigger 2 on timer E reset and counter roll-over(1)
Refer to ADC2TCRST description.

Bit 30 **ADC2TEC4**: ADC trigger 2 on timer E compare 4
Refer to ADC2TAC2 description.

Bit 29 **ADC2TEC3**: ADC trigger 2 on timer E compare 3
Refer to ADC2TAC2 description.

Bit 28 **ADC2TEC2**: ADC trigger 2 on timer E compare 2
Refer to ADC2TAC2 description.

Bit 27 **ADC2TDRST**: ADC trigger 2 on timer D reset and counter roll-over(1)
Refer to ADC2TCRST description.

Bit 26 **ADC2TDPER**: ADC trigger 2 on timer D period
Refer to ADC2TAPER description.

Bit 25 **ADC2TDC4**: ADC trigger 2 on timer D compare 4
Refer to ADC2TAC2 description.

Bit 24 **ADC2TFPER**: ADC trigger 2 on timer F period
Refer to ADC2TAPER description.

Bit 23 **ADC2TDC2**: ADC trigger 2 on timer D compare 2
Refer to ADC2TAC2 description.

Bit 22 **ADC2TCRST**: ADC trigger 2 on timer C reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer C reset and roll-over event, on ADC trigger 1 output.

Bit 21 **ADC2TCPER**: ADC trigger 2 on timer C period
Refer to ADC2TAPER description.

Bit 20 **ADC2TCC4**: ADC trigger 2 on timer C compare 4
Refer to ADC2TAC2 description.

Bit 19 **ADC2TFC4**: ADC trigger 2 on timer F compare 4
Refer to ADC2TAC2 description.

Bit 18 **ADC2TCC2**: ADC trigger 2 on timer C compare 2
Refer to ADC2TAC2 description.

Bit 17 **ADC2TBPER**: ADC trigger 2 on timer B period
Refer to ADC2TAPER description.

Bit 16 **ADC2TBC4**: ADC trigger 2 on timer B compare 4
Refer to ADC2TAC2 description.

Bit 15 **ADC2TFC3**: ADC trigger 2 on timer F compare 3
Refer to ADC2TAC2 description.

Bit 14 **ADC2TBC2**: ADC trigger 2 on timer B compare 2
Refer to ADC2TAC2 description.

Bit 13 **ADC2TAPER**: ADC trigger 2 on timer A period
This bit enables the generation of an ADC trigger upon timer A event, on ADC trigger 2 output.

Bit 12 **ADC2TAC4**: ADC trigger 2 on timer A compare 4
Refer to ADC2TAC2 description.

Bit 11 **ADC2TFC2**: ADC trigger 2 on timer F compare 2
Refer to ADC2TAC2 description.

Bit 10 **ADC2TAC2**: ADC trigger 2 on timer A compare 2
This bit enables the generation of an ADC trigger upon timer A compare 2, on ADC trigger 2 output.

Bit 9 **ADC2EEV10**: ADC trigger 2 on external event 10(1)
Refer to ADC2EEV6 description.

Bit 8 **ADC2EEV9**: ADC trigger 2 on external event 9(1)
Refer to ADC2EEV6 description.

Bit 7 **ADC2EEV8**: ADC trigger 2 on external event 8(1)
Refer to ADC2EEV6 description.

Bit 6 **ADC2EEV7**: ADC trigger 2 on external event 7(1)
Refer to ADC2EEV6 description.

Bit 5 **ADC2EEV6**: ADC trigger 2 on external event 6(1)
This bit enables the generation of an ADC trigger upon external event 6, on ADC trigger 2 output.

Bit 4 **ADC2MPER**: ADC trigger 2 on master period
This bit enables the generation of an ADC trigger upon master period event, on ADC trigger 2 output.

Bit 3 **ADC2MC4**: ADC trigger 2 on master compare 4
Refer to ADC2MC1 description.

Bit 2 **ADC2MC3**: ADC trigger 2 on master compare 3
Refer to ADC2MC1 description.

Bit 1 **ADC2MC2**: ADC trigger 2 on master compare 2
Refer to ADC2MC1 description.

Bit 0 **ADC2MC1**: ADC trigger 2 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 2 output.

1. These triggers are differing from HRTIM_ADC1R/HRTIM_ADC3R to HRTIM_ADC2R/HRTIM_ADC4R.

40.5.77 HRTIM ADC trigger 3 register (HRTIM_ADC3R)

Address offset: 0x744

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC3TEPER	ADC3TEC4	ADC3TEC3	ADC3TFRST	ADC3TDPER	ADC3TDC4	ADC3TDC3	ADC3TFPER	ADC3TCPER	ADC3TCC4	ADC3TCC3	ADC3TFC4	ADC3TBRST	ADC3TBPER	ADC3TBC4	ADC3TBC3
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3TFC3	ADC3TARST	ADC3TAPER	ADC3TAC4	ADC3TAC3	ADC3TFC2	ADC3EEV5	ADC3EEV4	ADC3EEV3	ADC3EEV2	ADC3EEV1	ADC3MPER	ADC3MC4	ADC3MC3	ADC3MC2	ADC3MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC3TEPER**: ADC trigger 3 on timer E period
Refer to ADC3TAPER description.

Bit 30 **ADC3TEC4**: ADC trigger 3 on timer E compare 4
Refer to ADC3TFC2 description.

Bit 29 **ADC3TEC3**: ADC trigger 3 on timer E compare 3
Refer to ADC3TFC2 description.

Bit 28 **ADC3TFRST**: ADC trigger 3 on timer F reset and counter roll-over(1)
Refer to ADC3TARST description.

Bit 27 **ADC3TDPER**: ADC trigger 3 on timer D period
Refer to ADC3TAPER description.

Bit 26 **ADC3TDC4**: ADC trigger 3 on timer D compare 4
Refer to ADC3TFC2 description.

Bit 25 **ADC3TDC3**: ADC trigger 3 on timer D compare 3
Refer to ADC3TFC2 description.

Bit 24 **ADC3TFPER**: ADC trigger 3 on timer F period
Refer to ADC3TAPER description.

Bit 23 **ADC3TCPER**: ADC trigger 3 on timer C period
Refer to ADC3TAPER description.

Bit 22 **ADC3TCC4**: ADC trigger 3 on timer C compare 4
Refer to ADC3TFC2 description.

Bit 21 **ADC3TCC3**: ADC trigger 3 on timer C compare 3
Refer to ADC3TFC2 description.

Bit 20 **ADC3TFC4**: ADC trigger 3 on timer F compare 4
Refer to ADC3TAC2 description.

Bit 19 **ADC3TBRST**: ADC trigger 3 on timer B reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer B reset and roll-over event, on ADC trigger 1 output.

Bit 18 **ADC3TBPER**: ADC trigger 3 on timer B period
Refer to ADC3TAPER description.

Bit 17 **ADC3TBC4**: ADC trigger 3 on timer B compare 4
Refer to ADC3TFC2 description.

Bit 16 **ADC3TBC3**: ADC trigger 3 on timer B compare 3
Refer to ADC3TFC2 description.

Bit 15 **ADC3TFC3**: ADC trigger 3 on timer F compare 3
Refer to ADC3TFC2 description.

Bit 14 **ADC3TARST**: ADC trigger 3 on timer A reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer A reset and roll-over event, on ADC trigger 1 output.

Bit 13 **ADC3TAPER**: ADC trigger 3 on timer A period
This bit enables the generation of an ADC trigger upon timer A period event, on ADC trigger 1 output.

Bit 12 **ADC3TAC4**: ADC trigger 3 on timer A compare 4
Refer to ADC3TFC2 description.

Bit 11 **ADC3TAC3**: ADC trigger 3 on timer A compare 3
Refer to ADC3TFC2 description.

Bit 10 **ADC3TFC2**: ADC trigger 3 on timer F compare 2
This bit enables the generation of an ADC trigger upon timer F compare 2 event, on ADC trigger 3 output.

Bit 9 **ADC3EEV5**: ADC trigger 3 on external event 5(1)
Refer to ADC3EEV1 description.

Bit 8 **ADC3EEV4**: ADC trigger 3 on external event 4(1)
Refer to ADC3EEV1 description.

Bit 7 **ADC3EEV3**: ADC trigger 3 on external event 3(1)
Refer to ADC3EEV1 description.

Bit 6 **ADC3EEV2**: ADC trigger 3 on external event 2(1)
Refer to ADC3EEV1 description.

Bit 5 **ADC3EEV1**: ADC trigger 3 on external event 1(1)
This bit enables the generation of an ADC trigger upon external event 1, on ADC trigger 3 output.

Bit 4 **ADC3MPER**: ADC trigger 3 on master period
This bit enables the generation of an ADC trigger upon master timer period event, on ADC trigger 3 output.

Bit 3 **ADC3MC4**: ADC trigger 3 on master compare 4
Refer to ADC3MC1 description.

Bit 2 **ADC3MC3**: ADC trigger 3 on master compare 3
Refer to ADC3MC1 description.

Bit 1 **ADC3MC2**: ADC trigger 3 on master compare 2
Refer to ADC3MC1 description.

Bit 0 **ADC3MC1**: ADC trigger 3 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 3 output.

1. These triggers are differing from HRTIM_ADC2R/HRTIM_ADC4R to HRTIM_ADC1R/HRTIM_ADC3R.

40.5.78 HRTIM ADC trigger 4 register (HRTIM_ADC4R)

Address offset: 0x748

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC4TERST	ADC4TEC4	ADC4TEC3	ADC4TEC2	ADC4TDRST	ADC4TDPER	ADC4TDC4	ADC4TFPER	ADC4TDC2	ADC4TCRST	ADC4TCPER	ADC4TCC4	ADC4TFC4	ADC4TCC2	ADC4TBPER	ADC4TBC4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC4TFC3	ADC4TBC2	ADC4TAPER	ADC4TAC4	ADC4TFC2	ADC4TAC2	ADC4EEV10	ADC4EEV9	ADC4EEV8	ADC4EEV7	ADC4EEV6	ADC4MPER	ADC4MC4	ADC4MC3	ADC4MC2	ADC4MC1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADC4TERST**: ADC trigger 4 on timer E reset and counter roll-over(1)
Refer to ADC4TCRST description.

Bit 30 **ADC4TEC4**: ADC trigger 4 on timer E compare 4
Refer to ADC4TAC2 description.

Bit 29 **ADC4TEC3**: ADC trigger 4 on timer E compare 3
Refer to ADC4TAC2 description.

Bit 28 **ADC4TEC2**: ADC trigger 4 on timer E compare 2
Refer to ADC4TAC2 description.

Bit 27 **ADC4TDRST**: ADC trigger 4 on timer D reset and counter roll-over(1)
Refer to ADC4TCRST description.

Bit 26 **ADC4TDPER**: ADC trigger 4 on timer D period
Refer to ADC4TAPER description.

Bit 25 **ADC4TDC4**: ADC trigger 4 on timer D compare 4
Refer to ADC4TAC2 description.

Bit 24 **ADC4TFPER**: ADC trigger 4 on timer F period
Refer to ADC4TAPER description.

Bit 23 **ADC4TDC2**: ADC trigger 2 on timer D compare 2
Refer to ADC4TAC2 description.

Bit 22 **ADC4TCRST**: ADC trigger 4 on timer C reset and counter roll-over(1)
This bit enables the generation of an ADC trigger upon timer C reset and roll-over event, on ADC trigger 1 output.

Bit 21 **ADC4TCPER**: ADC trigger 4 on timer C period
Refer to ADC4TAPER description.

Bit 20 **ADC4TCC4**: ADC trigger 4 on timer C compare 4
Refer to ADC4TAC2 description.

Bit 19 **ADC4TFC4**: ADC trigger 4 on timer F compare 4
Refer to ADC4TAC2 description.

Bit 18 **ADC4TCC2**: ADC trigger 4 on timer C compare 2
Refer to ADC4TAC2 description.

Bit 17 **ADC4TBPER**: ADC trigger 4 on timer B period
Refer to ADC4TAPER description.

Bit 16 **ADC4TBC4**: ADC trigger 4 on timer B compare 4
Refer to ADC4TAC2 description.

Bit 15 **ADC4TFC3**: ADC trigger 4 on timer F compare 3
Refer to ADC4TAC2 description.

Bit 14 **ADC4TBC2**: ADC trigger 4 on timer B compare 2
Refer to ADC4TAC2 description.

Bit 13 **ADC4TAPER**: ADC trigger 4 on timer A period
This bit enables the generation of an ADC trigger upon timer A event, on ADC trigger 2 output.

Bit 12 **ADC4TAC4**: ADC trigger 4 on timer A compare 4
Refer to ADC4TAC2 description.

Bit 11 **ADC4TFC2**: ADC trigger 4 on timer F compare 2
Refer to ADC4TAC2 description.

Bit 10 **ADC4TAC2**: ADC trigger 4 on timer A compare 2
This bit enables the generation of an ADC trigger upon timer A compare 2, on ADC trigger 2 output.

Bit 9 **ADC4EEV10**: ADC trigger 4 on external event 10(1)
Refer to ADC4EEV6 description.

Bit 8 **ADC4EEV9**: ADC trigger 4 on external event 9(1)
Refer to ADC4EEV6 description.

Bit 7 **ADC4EEV8**: ADC trigger 4 on external event 8(1)
Refer to ADC4EEV6 description.

Bit 6 **ADC4EEV7**: ADC trigger 4 on external event 7(1)
Refer to ADC4EEV6 description.

Bit 5 **ADC4EEV6**: ADC trigger 4 on external event 6(1)
This bit enables the generation of an ADC trigger upon external event 6, on ADC trigger 2 output.

Bit 4 **ADC4MPER**: ADC trigger 4 on master period
This bit enables the generation of an ADC trigger upon master period event, on ADC trigger 2 output.

Bit 3 **ADC4MC4**: ADC trigger 4 on master compare 4
Refer to ADC4MC1 description.

Bit 2 **ADC4MC3**: ADC trigger 4 on master compare 3
Refer to ADC4MC1 description.

Bit 1 **ADC4MC2**: ADC trigger 4 on master compare 2
Refer to ADC4MC1 description.

Bit 0 **ADC4MC1**: ADC trigger 4 on master compare 1
This bit enables the generation of an ADC trigger upon master compare 1 event, on ADC trigger 2 output.

1. These triggers are differing from HRTIM_ADC1R/HRTIM_ADC3R to HRTIM_ADC2R/HRTIM_ADC4R.

40.5.79 HRTIM DLL control register (HRTIM_DLLCR)

Address offset: 0x74C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALRTE[1:0]		CALEN	CAL
												rw	rw	rw	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **CALRTE[1:0]**: DLL Calibration rate

This defines the DLL calibration periodicity.
00: 1048576 * t_{HRTIM} (3.744 ms for f_{HRTIM} = 280 MHz)
01: 131072 * t_{HRTIM} (468 µs for f_{HRTIM} = 280 MHz)
10: 16384 * t_{HRTIM} (58.5 µs for f_{HRTIM} = 280 MHz)
11: 2048 * t_{HRTIM} (7.3 µs for f_{HRTIM} = 280 MHz)

Bit 1 **CALEN**: DLL Calibration Enable

This bit enables the periodic DLL calibration, as per CALRTE[1:0] bit setting. When CALEN bit is reset, the calibration can be started in single-shot mode with CAL bit.
0: Periodic calibration disabled
1: Calibration is performed periodically, as per CALRTE[1:0] setting

Note: CALEN must not be set simultaneously with CAL bit

Bit 0 **CAL**: DLL Calibration Start

This bit starts the DLL calibration process. It is write-only.
0: No calibration request
1: Calibration start

Note: CAL must not be set simultaneously with CALEN bit

40.5.80 HRTIM fault input register 1 (HRTIM_FLTINR1)

Address offset: 0x750

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4LCK	FLT4F[3:0]				FLT4SRC[0]	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC[0]	FLT3P	FLT3E
rwo	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT2LCK	FLT2F[3:0]				FLT2SRC[0]	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC[0]	FLT1P	FLT1E
rwo	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4LCK**: Fault 4 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 30:27 **FLT4F[3:0]**: Fault 4 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 26 FLT4SRC[0]: Fault 4 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 25 **FLT4P**: Fault 4 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 24 **FLT4E**: Fault 4 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 23 **FLT3LCK**: Fault 3 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 22:19 **FLT3F[3:0]**: Fault 3 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 18 FLT3SRC[0]: Fault 3 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 17 **FLT3P**: Fault 3 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 16 **FLT3E**: Fault 3 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 15 **FLT2LCK**: Fault 2 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 14:11 **FLT2F[3:0]**: Fault 2 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 10 FLT2SRC[0]: Fault 2 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 9 **FLT2P**: Fault 2 polarity
Refer to FLT2P description in HRTIM_FLTINR2 register.

Bit 8 **FLT2E**: Fault 2 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

Bit 7 **FLT1LCK**: Fault 1 lock
Refer to FLT5LCK description in HRTIM_FLTINR2 register.

Bits 6:3 **FLT1F[3:0]**: Fault 1 filter
Refer to FLT5F[3:0] description in HRTIM_FLTINR2 register.

Bit 2 **FLT1SRC[0]**: Fault 1 source bit 0
Refer to FLT5SRC[0] description in HRTIM_FLTINR2 register.

Bit 1 **FLT1P**: Fault 1 polarity
Refer to FLT5P description in HRTIM_FLTINR2 register.

Bit 0 **FLT1E**: Fault 1 enable
Refer to FLT5E description in HRTIM_FLTINR2 register.

40.5.81 HRTIM fault input register 2 (HRTIM_FLTINR2)

Address offset: 0x754

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		FLT8SRC[1]	FLT7SRC[1]	FLT6SRC[1]	FLT5SRC[1]	FLT4SRC[1]	FLT3SRC[1]	FLT2SRC[1]	FLT1SRC[1]
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT6LCK	FLT6F[3:0]				FLT6SRC[0]	FLT6P	FLT6E	FLT5LCK	FLT5F[3:0]				FLT5SRC[0]	FLT5P	FLT5E
rwo	rw	rw	rw	rw	rw	rw	rw	rwo	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **FLTSD[1:0]**: Fault sampling clock division

This bitfield indicates the division ratio between the timer clock frequency (f_{HRTIM}) and the fault signal sampling clock (f_{FLTS}) used by the digital filters.

00: $f_{FLTS}=f_{HRTIM}$

01: $f_{FLTS}=f_{HRTIM} / 2$

10: $f_{FLTS}=f_{HRTIM} / 4$

11: $f_{FLTS}=f_{HRTIM} / 8$

Note: This bitfield must be written prior to any of the FLTxE enable bits.

Bit 23 **FLT8SRC[1]**: Fault 8 source bit 1

Refer to FLT5SRC[0] description.

Bit 22 **FLT7SRC[1]**: Fault 7 source bit 1

Refer to FLT5SRC[0] description.

Bit 21 **FLT6SRC[1]**: Fault 6 source bit 1

Refer to FLT5SRC[0] description.

Bit 20 **FLT5SRC[1]**: Fault 5 source bit 1

Refer to FLT5SRC[0] description.

Bit 19 **FLT4SRC[1]**: Fault 4 source bit 1

Refer to FLT5SRC[0] description.

Bit 18 **FLT3SRC[1]**: Fault 3 source bit 1

Refer to FLT5SRC[0] description.

Bit 17 **FLT2SRC[1]**: Fault 2 source bit 1

Refer to FLT5SRC[0] description.

Bit 16 **FLT1SRC[1]**: Fault 1 source bit 1

Refer to FLT5SRC[0] description.

Bit 15 **FLT6LCK**: Fault 6 lock

Refer to FLT5LCK description.

Bits 14:11 **FLT6F[3:0]**: Fault 6 filter

Refer to FLT5F[3:0] description.

Bit 10 **FLT6SRC[0]**: Fault 6 source bit 0

Refer to FLT5SRC[0] description.

Bit 9 **FLT6P**: Fault 6 polarity
Refer to FLT5P description.

Bit 8 **FLT6E**: Fault 6 enable
Refer to FLT5E description.

Bit 7 **FLT5LCK**: Fault 5 lock
The FLT5LCK bit modifies the write attributes of the fault programming bit, so that they are protected against spurious write accesses.
This bit is write-once. Once it has been set, it cannot be modified till the next system reset.
0: FLT5E, FLT5P, FLT5SRC[1:0], FLT5F[3:0] and FLT5BLK bits are read/write.
1: FLT5E, FLT5P, FLT5SRC[1:0], FLT5F[3:0] and FLT5BLK bits cannot be written (read-only mode)

Bits 6:3 **FLT5F[3:0]**: Fault 5 filter
This bitfield defines the frequency used to sample FLT5 input and the length of the digital filter applied to FLT5. The digital filter is made of an event counter in which N events are needed to validate a transition on the output.
0000: No filter, FLT5 acts asynchronously
0001: $f_{SAMPLING} = f_{HRTIM}$, N = 2
0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4
0011: $f_{SAMPLING} = f_{HRTIM}$, N = 8
0100: $f_{SAMPLING} = f_{FLTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{FLTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{FLTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{FLTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{FLTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{FLTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{FLTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{FLTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{FLTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{FLTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{FLTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{FLTS}/32$, N = 8
Note: This bitfield can be written only when FLT5E enable bit is reset.
This bitfield cannot be modified when FLT5LOCK has been programmed.

Bit 2 **FLT5SRC[0]**: Fault 5 source bit 0
The FTL5SRC[1:0] bitfield selects the FAULT5 input source (refer to *Table 410* for connection details).
00: Fault 5 input is HRTIM_FLT5 input pin
01: Fault 5 input is connected to a COMPx output
10: Fault 5 input is EEV5_muxout input pin
11: hrtim_ext_trg5
Note: This bitfield can be written only when FLT5E enable bit is reset.

Bit 1 **FLT5P**: Fault 5 polarity
This bit selects the FAULT5 input polarity.
0: Fault 5 input is active low
1: Fault 5 input is active high
Note: This bitfield can be written only when FLT5E enable bit is reset.

Bit 0 **FLT5E**: Fault 5 enable
This bit enables the global FAULT5 input circuitry.
0: Fault 5 input disabled
1: Fault 5 input enabled

40.5.82 HRTIM burst DMA master timer update register (HRTIM_BDMUPR)

Address offset: 0x758

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MCMP4**: MCMP4R register update enable
Refer to MCR description.

Bit 8 **MCMP3**: MCMP3R register update enable
Refer to MCR description.

Bit 7 **MCMP2**: MCMP2R register update enable
Refer to MCR description.

Bit 6 **MCMP1**: MCMP1R register update enable
Refer to MCR description.

Bit 5 **MREP**: MREP register update enable
Refer to MCR description.

Bit 4 **MPER**: MPER register update enable
Refer to MCR description.

Bit 3 **MCNT**: MCNTR register update enable
Refer to MCR description.

Bit 2 **MDIER**: MDIER register update enable
Refer to MCR description.

Bit 1 **MICR**: MICR register update enable
Refer to MCR description.

Bit 0 **MCR**: MCR register update enable
This bit defines if the master timer MCR register is part of the list of registers to be updated by the burst DMA.
0: MCR register is not updated by burst DMA accesses
1: MCR register is updated by burst DMA accesses

40.5.83 HRTIM burst DMA timer x update register (HRTIM_BDTxUPR) (x = A to F)

Address offset: Block A: 0x75C

Address offset: Block B: 0x760

Address offset: Block C: 0x764

Address offset: Block D: 0x768

Address offset: Block E: 0x76C

Address offset: Block F: 0x774

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TIMxCODER	TIMxCOFER	TIMxCOTRGR	TIMxEESELR	TIMxEEFR3	TIMxCR2	TIMxFLTR	TIMxOUTR	TIMxCHPR	TIMxRSTR	TIMxEEFR2
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMxEEFR1	TIMxRST2R	TIMxSET2R	TIMxRST1R	TIMxSET1R	TIMxDTR	TIMxCMP4	TIMxCMP3	TIMxCMP2	TIMxCMP1	TIMxREP	TIMxPER	TIMxCNT	TIMxDIER	TIMxICR	TIMxCR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **TIMxCODER**: HRTIM_CODExR register update enable timer x
Refer to TIMxCR description.

Bit 25 **TIMxCOFER**: HRTIM_COFExR register update enable timer x
Refer to TIMxCR description.

Bit 24 **TIMxCOTRGR**: HRTIM_COTRGxR register update enable timer x
Refer to TIMxCR description.

Bit 23 **TIMxEESELR**: HRTIM_EESELxR register update enable timer x
Refer to TIMxCR description.

Bit 22 **TIMxEEFR3**: HRTIM_EEFxR3 register update enable timer x
Refer to TIMxCR description.

Bit 21 **TIMxCR2**: HRTIM_TIMxCR2 register update enable timer x
Refer to TIMxCR description.

Bit 20 **TIMxFLTR**: HRTIM_FLTxR register update enable timer x
Refer to TIMxCR description.

Bit 19 **TIMxOUTR**: HRTIM_OUTxR register update enable timer x
Refer to TIMxCR description.

Bit 18 **TIMxCHPR**: HRTIM_CHPxR register update enable timer x
Refer to TIMxCR description.

Bit 17 **TIMxRSTR**: HRTIM_RSTxR register update enable timer x
Refer to TIMxCR description.

Bit 16 **TIMxEEFR2**: HRTIM_EEFxR2 register update enable timer x
Refer to TIMxCR description.

Bit 15 **TIMxEEFR1**: HRTIM_EEFxR1 register update enable timer x
Refer to TIMxCR description.

Bit 14 **TIMxRST2R**: HRTIM_RST2xR register update enable timer x
Refer to TIMxCR description.

Bit 13 **TIMxSET2R**: HRTIM_SET2xR register update enable timer x
Refer to TIMxCR description.

Bit 12 **TIMxRST1R**: HRTIM_RST1xR register update enable timer x
Refer to TIMxCR description.

Bit 11 **TIMxSET1R**: HRTIM_SET1xR register update enable timer x
Refer to TIMxCR description.

Bit 10 **TIMxDTR**: HRTIM_DTxR register update enable timer x
Refer to TIMxCR description.

Bit 9 **TIMxCMP4**: HRTIM_CMP4xR register update enable timer x
Refer to TIMxCR description.

Bit 8 **TIMxCMP3**: HRTIM_CMP3xR register update enable timer x
Refer to TIMxCR description.

Bit 7 **TIMxCMP2**: HRTIM_CMP2xR register update enable timer x
Refer to TIMxCR description.

Bit 6 **TIMxCMP1**: HRTIM_CMP1xR register update enable timer x
Refer to TIMxCR description.

Bit 5 **TIMxREP**: HRTIM_REPxR register update enable timer x
Refer to TIMxCR description.

Bit 4 **TIMxPER**: HRTIM_PERxR register update enable timer x
Refer to TIMxCR description.

Bit 3 **TIMxCNT**: HRTIM_CNTxR register update enable timer x
Refer to TIMxCR description.

Bit 2 **TIMxDIER**: HRTIM_TIMxDIER register update enable timer x
Refer to TIMxCR description.

Bit 1 **TIMxICR**: HRTIM_TIMxICR register update enable timer x
Refer to TIMxCR description.

Bit 0 **TIMxCR**: HRTIM_TIMxCR register update enable timer x
This bit defines if the master timer MCR register is part of the list of registers to be updated by the burst DMA.
0: HRTIM_TIMxCR register is not updated by burst DMA accesses
1: HRTIM_TIMxCR register is updated by burst DMA accesses

40.5.84 HRTIM burst DMA data register (HRTIM_BDMADR)

Address offset: 0x770

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BDMADR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDMADR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **BDMADR[31:0]**: Burst DMA data register

Write accesses to this register triggers:

– the copy of the data value into the registers enabled in BDTxUPR and BDMUPR register bits

– the increment of the register pointer to the next location to be filled

40.5.85 HRTIM ADC extended trigger register (HRTIM_ADCER)

Address offset: 0x778

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ADC10TRG[4:0]					ADC9TRG[4:0]					ADC8TRG[4:0]				
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADC7TRG[4:0]					ADC6TRG[4:0]					ADC5TRG[4:0]				
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **ADC10TRG[4:0]**: ADC trigger 10 selection
This bit selects the ADC trigger 10 source.
Refer to ADC6TRG[4:0] description.

Bits 25:21 **ADC9TRG[4:0]**: ADC trigger 9 selection
This bit selects the ADC trigger 9 source.
Refer to ADC5TRG[4:0] description.

Bits 20:16 **ADC8TRG[4:0]**: ADC trigger 8 selection
This bit selects the ADC trigger 8 source.
Refer to ADC6TRG[4:0] description.

Bit 15 Reserved, must be kept at reset value.

Bits 14:10 **ADC7TRG[4:0]**: ADC trigger 7 selection
This bit selects the ADC trigger 7 source.
Refer to ADC5TRG[4:0] description.

Bits 9:5 **ADC6TRG[4:0]**: ADC trigger 6 selection
This bit selects the ADC trigger 6 source.
0: Trigger on master compare 1
1: Trigger on master compare 2
2: Trigger on master compare 3
3: Trigger on master compare 4
4: Trigger on master period
5: Trigger on external event 6
6: Trigger on external event 7
7: Trigger on external event 8
8: Trigger on external event 9
9: Trigger on external event 10
10: Trigger on timer A compare 2
11: Trigger on timer A compare 4
12: Trigger on timer A period
13: Trigger on timer B compare 2
14: Trigger on timer B compare 4
15: Trigger on timer B period
16: Trigger on timer C compare 2
17: Trigger on timer C compare 4
18: Trigger on timer C period
19: Trigger on timer C reset and counter roll-over
20: Trigger on timer D compare 2
21: Trigger on timer D compare 4
22: Trigger on timer D period
23: Trigger on timer D reset and counter roll-over
24: Trigger on timer E compare 2
25: Trigger on timer E compare 3
26: Trigger on timer E compare 4
27: Trigger on timer E reset and counter roll-over
28: Trigger on timer F compare 2
29: Trigger on timer F compare 3
30: Trigger on timer F compare 4
31: Trigger on timer F period

Bits 4:0 **ADC5TRG[4:0]**: ADC trigger 5 selection

This bit selects the ADC trigger 5 source.

0: Trigger on master compare 1
1: Trigger on master compare 2
2: Trigger on master compare 3
3: Trigger on master compare 4
4: Trigger on master period
5: Trigger on external event 1
6: Trigger on external event 2
7: Trigger on external event 3
8: Trigger on external event 4
9: Trigger on external event 5
10: Trigger on timer A compare 3
11: Trigger on timer A compare 4
12: Trigger on timer A period
13: Trigger on timer A reset and counter roll-over
14: Trigger on timer B compare 3
15: Trigger on timer B compare 4
16: Trigger on timer B period
17: Trigger on timer B reset and counter roll-over
18: Trigger on timer C compare 3
19: Trigger on timer C compare 4
20: Trigger on timer C period
21: Trigger on timer D compare 3
22: Trigger on timer D compare 4
23: Trigger on timer D period
24: Trigger on timer E compare 3
25: Trigger on timer E compare 4
26: Trigger on timer E period
27: Trigger on timer F compare 2
28: Trigger on timer F compare 3
29: Trigger on timer F compare 4
30: Trigger on timer F period
31: Trigger on timer F reset and counter roll-over

40.5.86 HRTIM ADC trigger update register (HRTIM_ADCUR)

Address offset: 0x77C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AD10USRC[2:0]			Res.	AD9USRC[2:0]		
									rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AD8USRC[2:0]			Res.	AD7USRC[2:0]			Res.	AD6USRC[2:0]			Res.	AD5USRC[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **AD10USRC[2:0]**: ADC trigger 10 update source
Refer to AD5USRC[2:0] description.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **AD9USRC[2:0]**: ADC trigger 9 update source
Refer to AD5USRC[2:0] description.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **AD8USRC[2:0]**: ADC trigger 8 update source
Refer to AD5USRC[2:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **AD7USRC[2:0]**: ADC trigger 7 update source
Refer to AD5USRC[2:0] description.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **AD6USRC[2:0]**: ADC trigger 6 update source
Refer to AD5USRC[2:0] description.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **AD5USRC[2:0]**: ADC trigger 5 update source
These bits define the source which triggers the update of the ADC5TRG[4:0] bitfield in the HRTIM_ADCER register (transfer from preload to active register). It only defines the source timer. The precise condition is defined within the timer itself, with the BRSTDMA[1:0] bitfield in HRTIM_MCR or the UPDGAT[3:0] bitfield in HRTIM_TIMxCR register.
000: Master timer
001: Timer A
010: Timer B
011: Timer C
100: Timer D
101: Timer E
110: Timer F
111: Reserved

40.5.87 HRTIM ADC post scaler register 1 (HRTIM_ADCPS1)

Address offset: 0x780

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC5PSC[4:0]					Res.	ADC4PSC[4:0]					Res.	ADC3PSC[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3PSC[3:0]				Res.	ADC2PSC[4:0]					Res.	ADC1PSC[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADC5PSC[4:0]**: ADC 5 post scaler
Refer to ADC1PSC[4:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **ADC4PSC[4:0]**: ADC 4 post scaler
Refer to ADC1PSC[4:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **ADC3PSC[4:0]**: ADC 3 post scaler
Refer to ADC1PSC[4:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **ADC2PSC[4:0]**: ADC 2 post scaler
Refer to ADC1PSC[4:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **ADC1PSC[4:0]**: ADC 1 post scaler
This bit selects the ADC 1 Post scaler ratio.

40.5.88 HRTIM ADC post scaler register 2 (HRTIM_ADCPS2)

Address offset: 0x784

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADC10PSC[4:0]					Res.	ADC9PSC[4:0]					Res.	ADC8PSC[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC8PSC[3:0]				Res.	ADC7PSC[4:0]					Res.	ADC6PSC[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADC10PSC[4:0]**: ADC 10 post scaler
Refer to ADC1PSC[4:0] description.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **ADC9PSC[4:0]**: ADC 9 post scaler
Refer to ADC1PSC[4:0] description.

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **ADC8PSC[4:0]**: ADC 8 post scaler
Refer to ADC1PSC[4:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **ADC7PSC[4:0]**: ADC 7 post scaler
Refer to ADC1PSC[4:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **ADC6PSC[4:0]**: ADC 6 post scaler
Refer to ADC1PSC[4:0] description.

40.5.89 HRTIM fault input register 3 (HRTIM_FLTINR3)

Address offset: 0x788

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT4RSTM	FLT4CRES	FLT4CNT[3:0]				FLT4BLKS	FLT4BLKE	FLT3RSTM	FLT3CRES	FLT3CNT[3:0]				FLT3BLKS	FLT3BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT2RSTM	FLT2CRES	FLT2CNT[3:0]				FLT2BLKS	FLT2BLKE	FLT1RSTM	FLT1CRES	FLT1CNT[3:0]				FLT1BLKS	FLT1BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT4RSTM**: Fault 4 reset mode
Refer to FLT5RSTM description.

Bit 30 **FLT4CRES**: Fault 4 counter reset
Refer to FLT5CRES description.

Bits 29:26 **FLT4CNT[3:0]**: Fault 4 counter
Refer to FLT5CNT description.

Bit 25 **FLT4BLKS**: Fault 4 blanking source
Refer to FLT5BLKS description.

Bit 24 **FLT4BLKE**: Fault 4 blanking enable
Refer to FLT5BLKE description.

Bit 23 **FLT3RSTM**: Fault 3 reset mode
Refer to FLT5RSTM description.

Bit 22 **FLT3CRES**: Fault 3 counter reset
Refer to FLT5CRES description.

Bits 21:18 **FLT3CNT[3:0]**: Fault 3 counter
Refer to FLT5CNT description.

Bit 17 **FLT3BLKS**: Fault 3 blanking source
Refer to FLT5BLKS description.

Bit 16 **FLT3BLKE**: Fault 3 blanking enable
Refer to FLT5BLKE description.

Bit 15 **FLT2RSTM**: Fault 2 reset mode
Refer to FLT5RSTM description.

Bit 14 **FLT2CRES**: Fault 2 counter reset
Refer to FLT5CRES description.

Bits 13:10 **FLT2CNT[3:0]**: Fault 2 counter
Refer to FLT5CNT description.

Bit 9 **FLT2BLKS**: Fault 2 blanking source
Refer to FLT5BLKS description.

Bit 8 **FLT2BLKE**: Fault 2 blanking enable
Refer to FLT5BLKE description.

Bit 7 **FLT1RSTM**: Fault 1 reset mode
Refer to FLT5RSTM description.

Bit 6 **FLT1CRES**: Fault 1 counter reset
Refer to FLT5CRES description.

Bits 5:2 **FLT1CNT[3:0]**: Fault 1 counter
Refer to FLT5CNT description.

Bit 1 **FLT1BLKS**: Fault 1 blanking source
Refer to FLT5BLKS description.

Bit 0 **FLT1BLKE**: Fault 1 blanking enable
Refer to FLT5BLKE description.

40.5.90 HRTIM fault input register 4 (HRTIM_FLTINR4)

Address offset: 0x78C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLT8RSTM	FLT8CRES	FLT8CNT[3:0]				FLT8BLKS	FLT8BLKE	FLT7RSTM	FLT7CRES	FLT7CNT[3:0]				FLT7BLKS	FLT7BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT6RSTM	FLT6CRES	FLT6CNT[3:0]				FLT6BLKS	FLT6BLKE	FLT5RSTM	FLT5CRES	FLT5CNT[3:0]				FLT5BLKS	FLT5BLKE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLT8RSTM**: Fault 8 reset mode
Refer to FLT5RSTM description.

Bit 30 **FLT8CRES**: Fault 8 counter reset
Refer to FLT5CRES description.

Bits 29:26 **FLT8CNT[3:0]**: Fault 8 counter
Refer to FLT5CNT description.

Bit 25 **FLT8BLKS**: Fault 8 blanking source
Refer to FLT5BLKS description.

Bit 24 **FLT8BLKE**: Fault 8 blanking enable
Refer to FLT5BLKE description.

Bit 23 **FLT7RSTM**: Fault 7 reset mode
Refer to FLT5RSTM description.

Bit 22 **FLT7CRES**: Fault 7 counter reset
Refer to FLT5CRES description.

Bits 21:18 **FLT7CNT[3:0]**: Fault 7 counter
Refer to FLT5CNT description.

Bit 17 **FLT7BLKS**: Fault 7 blanking source
Refer to FLT5BLKS description.

Bit 16 **FLT7BLKE**: Fault 7 blanking enable
Refer to FLT5BLKE description.

Bit 15 **FLT6RSTM**: Fault 6 reset mode
Refer to FLT5RSTM description.

Bit 14 **FLT6CRES**: Fault 6 counter reset
Refer to FLT5CRES description.

Bits 13:10 **FLT6CNT[3:0]**: Fault 6 counter
Refer to FLT5CNT description.

Bit 9 **FLT6BLKS**: Fault 6 blanking source
Refer to FLT5BLKS description.

Bit 8 **FLT6BLKE**: Fault 6 blanking enable
Refer to FLT5BLKE description.

Bit 7 **FLT5RSTM**: Fault 5 reset mode

This bit selects the FAULT5 counter reset mode

0: Fault 5 counter is reset on each reset / roll-over event

1: Fault 5 counter is reset on each reset / roll-over event only if no fault occurred during last counting period.

Note: This bitfield can be written only when FLT5E enable bit is reset.

Bit 6 **FLT5CRES**: Fault 5 counter reset

This bit resets the FAULT5 counter. It is set by software and reset by hardware.

0: No action

1: Fault 5 counter is reset

Bits 5:2 **FLT5CNT[3:0]**: Fault 5 counter

This bitfield selects the FAULT5 counter threshold. A fault is considered valid when the number of events is equal to the (FLT5CNT[3:0]+1) value.

Bit 1 **FLT5BLKS**: Fault 5 blanking source

The FTL5BLKS bit selects the FAULT5 blanking source (refer to *Table 441* for details).

0: Fault 5 reset-aligned blanking window

1: Fault 5 Moving blanking window

Note: This bitfield can be written only when FLT5E enable bit is reset.

Bit 0 **FLT5BLKE**: Fault 5 blanking enable

The FTL5BLKE bit selects the FAULT5 blanking mode. The blanking source is defined by the FLT5BLKS bit.

0: No blanking on fault 5

1: Fault 5 blanking mode

Note: This bitfield can be written only when FLT5E enable bit is reset

40.5.91 HRTIM fault input register 5 (HRTIM_FLTINR5)

Address offset: 0x790

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLT8LCK	FLT8F[3:0]				FLT8SRC[0]	FLT8P	FLT8E	FLT7LCK	FLT7F[3:0]				FLT7SRC[0]	FLT7P	FLT7E
rwo	rw	rw	rw	rw	rw	rw	rw	rwo	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLT8LCK**: Fault 8 lock
Refer to FLT7LCK description.

Bits 14:11 **FLT8F[3:0]**: Fault 8 filter
Refer to FLT7F[3:0] description.

Bit 10 **FLT8SRC[0]**: Fault 8 source bit 0
Refer to FLT7SRC[0] description.

Bit 9 **FLT8P**: Fault 8 polarity
Refer to FLT7P description.

Bit 8 **FLT8E**: Fault 8 enable
Refer to FLT7E description.

Bit 7 **FLT7LCK**: Fault 7 lock
The FLT7LCK bit modifies the write attributes of the fault programming bit, so that they are protected against spurious write accesses.
This bit is write-once. Once it has been set, it cannot be modified till the next system reset.
0: FLT7E, FLT7P, FLT7SRC[1:0], FLT7F[3:0] and FLT7BLK bits are read/write.
1: FLT7E, FLT7P, FLT7SRC[1:0], FLT7F[3:0] and FLT7BLK bits cannot be written (read-only mode)

Bits 6:3 **FLT7F[3:0]**: Fault 7 filter

This bitfield defines the frequency used to sample FLT7 input and the length of the digital filter applied to FLT7. The digital filter is made of an event counter in which N events are needed to validate a transition on the output.

0000: No filter, FLT7 acts asynchronously
0001: $f_{SAMPLING} = f_{HRTIM}$, N = 2
0010: $f_{SAMPLING} = f_{HRTIM}$, N = 4
0011: $f_{SAMPLING} = f_{HRTIM}$, N = 8
0100: $f_{SAMPLING} = f_{FLTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{FLTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{FLTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{FLTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{FLTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{FLTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{FLTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{FLTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{FLTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{FLTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{FLTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{FLTS}/32$, N = 8

Note: This bitfield can be written only when FLT7E enable bit is reset.
This bitfield cannot be modified when FLT7LOCK has been programmed.

Bit 2 **FLT7SRC[0]**: Fault 7 source bit 0

The FTL7SRC[1:0] bitfield selects the FAULT7 input source (refer to *Table 410* for connection details).

00: Fault 7 input is HRTIM_FLT7 input pin
01: Fault 7 input is connected to a COMPx output
10: Fault 7 input is EEV7_muxout input pin
01: hrtim_ext_trg7

Note: This bitfield can be written only when FLT7E enable bit is reset.

Bit 1 **FLT7P**: Fault 7 polarity

This bit selects the FAULT7 input polarity.

0: Fault 7 input is active low
1: Fault 7 input is active high

Note: This bitfield can be written only when FLT7E enable bit is reset.

Bit 0 **FLT7E**: Fault 7 enable

This bit enables the global FAULT7 input circuitry.

0: Fault 7 input disabled
1: Fault 7 input enabled

40.5.92 HRTIM timer external event control register 4 (HRTIM_EECR4)

Address offset: 0x7A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE13FAST	EE13SNS[1]
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE13SNS[0]	EE13POL	EE13SRC[1:0]		EE12FAST	EE12SNS[1:0]		EE12POL	EE12SRC[1:0]		EE11FAST	EE11SNS[1:0]		EE11POL	EE11SRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **EE13FAST**: External event 3 fast mode
Refer to EE11FAST description.

Bits 16:15 **EE13SNS[1:0]**: External event 3 sensitivity
Refer to EE11SNS[1:0] description.

Bit 14 **EE13POL**: External event 3 polarity
Refer to EE11POL description.

Bits 13:12 **EE13SRC[1:0]**: External event 3 source
This bitfield is combined with EE13SRC[2] bit. Refer to EE11SRC[1:0] description.

Bit 11 **EE12FAST**: External event 2 fast mode
Refer to EE11FAST description.

Bits 10:9 **EE12SNS[1:0]**: External event 2 sensitivity
Refer to EE11SNS[1:0] description.

Bit 8 **EE12POL**: External event 2 polarity
Refer to EE11POL description.

Bits 7:6 **EE12SRC[1:0]**: External event 2 source
This bitfield is combined with EE12SRC[2] bit. Refer to EE11SRC[1:0] description.

Bit 5 **EE11FAST**: External event 11 fast mode
0: External event 11 is re-synchronized by the HRTIM logic before acting on outputs, which adds a f_{HRTIM} clock-related latency
1: External event 11 is acting asynchronously on outputs (low latency mode)
Note: This bit must not be modified once the counter in which the event is used is enabled (TxCEN bit set).

Bits 4:3 **EE11SNS[1:0]**: External event 11 sensitivity
00: On active level defined by EE11POL bit
01: Rising edge, whatever EE11POL bit value
10: Falling edge, whatever EE11POL bit value
11: Both edges, whatever EE11POL bit value

Bit 2 **EE11POL**: External event 11 polarity
This bit is only significant if EE11SNS[1:0] = 00.
0: External event is active high
1: External event is active low

Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE11FAST bit.

Bits 1:0 **EE11SRC[1:0]**: External event 11 source
This bitfield selects the External event 11 source. This bitfield is combined with EE11SRC[2] bit. See *Table 411* for details.
000:hrtim_eev11_1
001:hrtim_eev11_2
010:hrtim_eev11_3
011:hrtim_eev11_4
100:hrtim_eev11_5
101:hrtim_eev11_6
110:hrtim_eev11_7
111:hrtim_eev11_8

Note: This parameter cannot be changed once the timer x is enabled. It must be configured prior to setting EE11FAST bit.

40.5.93 HRTIM timer external event control register 5 (HRTIM_EECR5)

Address offset: 0x7A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE16SNS[1]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE16SNS[0]	EE16POL	EE16SRC[1:0]		Res.	EE15SNS[1:0]		EE15POL	EE15SRC[1:0]		Res.	EE14SNS[1:0]		EE14POL	EE14SRC[1:0]	
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:15 **EE16SNS[1:0]**: External event 16 sensitivity
Refer to EE14SNS[1:0] description.

Bit 14 **EE16POL**: External event 16 polarity
Refer to EE14POL description.

Bits 13:12 **EE16SRC[1:0]**: External event 16 source
This bitfield is combined with EE16SRC[2] bit. Refer to EE14SRC[1:0] description.

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **EE15SNS[1:0]**: External event 15 sensitivity
Refer to EE14SNS[1:0] description.

Bit 8 **EE15POL**: External event 15 polarity
Refer to EE14POL description.

Bits 7:6 **EE15SRC[1:0]**: External event 15 source
This bitfield is combined with EE15SRC[2] bit. Refer to EE14SRC[1:0] description.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **EE14SNS[1:0]**: External event 14 sensitivity
00: On active level defined by EE14POL bit
01: Rising edge, whatever EE14POL bit value
10: Falling edge, whatever EE14POL bit value
11: Both edges, whatever EE14POL bit value

Bit 2 **EE14POL**: External event 14 polarity
This bit is only significant if EE14SNS[1:0] = 00.
0: External event is active high
1: External event is active low

Bits 1:0 **EE14SRC[1:0]**: External event 14 source
This bitfield is combined with EE14SRC[2] bit. This bitfield selects the External event 14 source. See *Table 411* for details.
000:hrtim_eev14_1
001:hrtim_eev14_2
010:hrtim_eev14_3
011:hrtim_eev14_4
100:hrtim_eev14_5
101:hrtim_eev14_6
110:hrtim_eev14_7
111:hrtim_eev14_8

40.5.94 HRTIM timer external event control register 6 (HRTIM_EECR6)

Address offset: 0x7A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EE16F[3:0]				Res.	Res.	EE15F[3:0]				Res.	Res.	EE14F[3:0]			
rw	rw	rw	rw			rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **EE16F[3:0]**: External event 16 filter
Refer to EE14F[3:0] description.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **EE15F[3:0]**: External event 15 filter
Refer to EE14F[3:0] description.

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **EE14F[3:0]**: External event 14 filter
This bitfield defines the frequency used to sample external event 6 input and the length of the digital filter applied to EEV14. The digital filter is made of a counter in which N valid samples are needed to validate a transition on the output.
0000: Filter disabled
0001: $f_{SAMPLING} = f_{HRTIM}$, N=2
0010: $f_{SAMPLING} = f_{HRTIM}$, N=4
0011: $f_{SAMPLING} = f_{HRTIM}$, N=8
0100: $f_{SAMPLING} = f_{EEVS}/2$, N=6
0101: $f_{SAMPLING} = f_{EEVS}/2$, N=8
0110: $f_{SAMPLING} = f_{EEVS}/4$, N=6
0111: $f_{SAMPLING} = f_{EEVS}/4$, N=8
1000: $f_{SAMPLING} = f_{EEVS}/8$, N=6
1001: $f_{SAMPLING} = f_{EEVS}/8$, N=8
1010: $f_{SAMPLING} = f_{EEVS}/16$, N=5
1011: $f_{SAMPLING} = f_{EEVS}/16$, N=6
1100: $f_{SAMPLING} = f_{EEVS}/16$, N=8
1101: $f_{SAMPLING} = f_{EEVS}/32$, N=5
1110: $f_{SAMPLING} = f_{EEVS}/32$, N=6
1111: $f_{SAMPLING} = f_{EEVS}/32$, N=8

40.5.95 HRTIM timer external event control register 7 (HRTIM_EECR7)

Address offset: 0x7AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	EE16SRC[2]	Res.	EE15SRC[2]	Res.	EE14SRC[2]	Res.	EE13SRC[2]	Res.	EE12SRC[2]	Res.	EE11SRC[2]	Res.	EE10SRC[2]	Res.	EE9SRC[2]
	rw		rw		rw		rw		rw		rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EE8SRC[2]	Res.	EE7SRC[2]	Res.	EE6SRC[2]	Res.	EE5SRC[2]	Res.	EE4SRC[2]	Res.	EE3SRC[2]	Res.	EE2SRC[2]	Res.	EE1SRC[2]
	rw		rw		rw		rw		rw		rw		rw		rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **EE16SRC[2]**: External event 16 source - bit 2
Refer to EE16SRC[1:0] bitfield description

Bit 29 Reserved, must be kept at reset value.

Bit 28 **EE15SRC[2]**: External event 15 source - bit 2
Refer to EE15SRC[1:0] bitfield description

Bit 27 Reserved, must be kept at reset value.

Bit 26 **EE14SRC[2]**: External event 14 source - bit 2
Refer to EE14SRC[1:0] bitfield description

Bit 25 Reserved, must be kept at reset value.

Bit 24 **EE13SRC[2]**: External event 13 source - bit 2
Refer to EE13SRC[1:0] bitfield description

Bit 23 Reserved, must be kept at reset value.

Bit 22 **EE12SRC[2]**: External event 12 source - bit 2
Refer to EE12SRC[1:0] bitfield description

Bit 21 Reserved, must be kept at reset value.

Bit 20 **EE11SRC[2]**: External event 11 source - bit 2
Refer to EE11SRC[1:0] bitfield description

Bit 19 Reserved, must be kept at reset value.

Bit 18 **EE10SRC[2]**: External event 10 source - bit 2
Refer to EE10SRC[1:0] bitfield description

Bit 17 Reserved, must be kept at reset value.

Bit 16 **EE9SRC[2]**: External event 9 source - bit 2
Refer to EE9SRC[1:0] bitfield description

Bit 15 Reserved, must be kept at reset value.

Bit 14 **EE8SRC[2]**: External event 8 source - bit 2
Refer to EE8SRC[1:0] bitfield description

Bit 13 Reserved, must be kept at reset value.

Bit 12 **EE7SRC[2]**: External event 7 source - bit 2
Refer to EE7SRC[1:0] bitfield description

Bit 11 Reserved, must be kept at reset value.

Bit 10 **EE6SRC[2]**: External event 6 source - bit 2
Refer to EE6SRC[1:0] bitfield description

Bit 9 Reserved, must be kept at reset value.

Bit 8 **EE5SRC[2]**: External event 5 source - bit 2
Refer to EE5SRC[1:0] bitfield description

Bit 7 Reserved, must be kept at reset value.

Bit 6 **EE4SRC[2]**: External event 4 source - bit 2
Refer to EE4SRC[1:0] bitfield description

Bit 5 Reserved, must be kept at reset value.

Bit 4 **EE3SRC[2]**: External event 3 source - bit 2
Refer to EE3SRC[1:0] bitfield description

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EE2SRC[2]**: External event 2 source - bit 2
Refer to EE2SRC[1:0] bitfield description

Bit 1 Reserved, must be kept at reset value.

Bit 0 **EE1SRC[2]**: External event 1 source - bit 2
Refer to EE1SRC[1:0] bitfield description

40.5.96 HRTIM control register 3 (HRTIM_CR3)

Address offset: 0x7C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTFUDISA	RTEUDISA	RTDUDISA	RTCUDISA	RTBUDISA	RTAUDISA	RMUDISA
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **RTFUDISA**: Remote timer F update disable authorization
Refer to RTAUDISA description.

Bit 5 **RTEUDISA**: Remote timer E update disable authorization
Refer to RTAUDISA description

Bit 4 **RTDUDISA**: Remote timer D update disable authorization
Refer to RTAUDISA description.

Bit 3 **RTCUDISA**: Remote timer C update disable authorization
Refer to RTAUDISA description.

Bit 2 **RTBUDISA**: Remote timer B update disable authorization
Refer to RTAUDISA description.

Bit 1 **RTAUDISA**: Remote timer A update disable authorization
This bit is set and cleared by software to authorize another HRTIMER to disable Timer A's update events.
0: Remote update disable requests are ignored.
1: Remote update disable requests authorized.

Bit 0 **RMUDISA**: Remote master update disable authorization
This bit is set and cleared by software to authorize another HRTIMER to disable master timer's update events.
0: Remote update disable requests are ignored.
1: Remote update disable requests authorized.

40.5.97 HRTIM option register (HRTIM_OR)

Address offset: 0x7FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSYNCSEL	UPDENSEL	BMTRGSEL
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **INSYNCSEL**: Synchronization Input Selection

This bit control a multiplexer for selecting the signal connected to the hrtim_in_sync3 input. See *Table 415: Internal synchronization source* for details.

0: hrtim_in_sync3 is connected to HRTIMx_SCIN

1: hrtim_in_sync3 is connected to play1_out15

Bit 1 **UPDENSEL**: Update Enable Selection

This bit control a multiplexer for selecting the signal connected to the hrtim_upd_en1 input. See *Table 412: Update enable inputs and sources* for details.

0: hrtim_upd_en1 is connected to tim10_oc1

1: hrtim_upd_en1 is connected to play1_out14

Bit 0 **BMTRGSEL**: Update Enable Selection

This bit control a multiplexer for selecting the signal connected to the hrtim_bm_trg input. See *Table 414: Burst mode trigger source* for details.

0: hrtim_bm_trg is connected to tim26_trgo

1: hrtim_bm_trg is connected to play1_out13

40.5.98 HRTIM register map

The tables below summarize the HRTIM registers mapping.

Table 447. HRTIM Register map and reset values – master timer

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**HRTIM_MCR**	BRSTDMA[1:0]		MREPU	Res.	PREEN	DACSYNC[1:0]		Res.	Res.	TFCEN	TECEN	TDCEN	TCCEN	TBCEN	TACEN	MCEN	SYNCSRC[1:0]		SYNCOUT[1:0]		SYNCSTRTM	SYNCRSTM	SYNCIN[1:0]		INTLVD[1:0]		HALF	RETRIG	CONT	CKPSC[2:0]		
	Reset value	0	0	0		0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**HRTIM_MISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPD	SYNC	MREP	MCMP4	MCMP3	MCMP2	MCMP1
	Reset value																										0	0	0	0	0	0	0
0x008	**HRTIM_MICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDC	SYNCC	MREPC	MCMP4C	MCMP3C	MCMP2C	MCMP1C
	Reset value																										0	0	0	0	0	0	0
0x00C	**HRTIM_MDIER**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDDE	SYNCDE	MREPDE	MCMP4DE	MCMP3DE	MCMP2DE	MCMP1DE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUPDIE	SYNCIE	MREPIE	MCMP4IE	MCMP3IE	MCMP2IE	MCMP1IE
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x010	**HRTIM_MCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**HRTIM_MPER**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPER[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x018	**HRTIM_MREP**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MREP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	**HRTIM_MCMP1R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	Reserved																																
0x024	**HRTIM_MCMP2R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**HRTIM_MCMP3R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**HRTIM_MCMP4R**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This register can be preloaded (see *Table 433 on page 1660*).

Table 448. HRTIM register map and reset values – TIMx (x= A..F)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100 (TIMA) 0x200 (TIMB) 0x300 (TIMC) 0x400 (TIMD) 0x500 (TIME) 0x600 (TIMF)	**HRTIM_ TIMxCR**	UPDGAT [3:0]				PREEN	DACSYNC[1:0]		MSTU	TEU	TDU	TCU	TBU	TAU	TxRSTU	TxREPU	TFU	DELCMP4[1:0]		DELCMP2[1:0]		SYNCSTRTx	SYNCRSTx	RSYNCU	INTLVD[1:0]		PSHPLL	HALF	RETRIG	CONT	CKPSCx[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104 (TIMA) 0x204 (TIMB) 0x304 (TIMC) 0x404 (TIMD) 0x504 (TIME) 0x604 (TIMF)	**HRTIM_ TIMxISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COTS	COS	O2CPY	O1CPY	O2STAT	O1STAT	IPPSTAT	CPPSTAT	CORDE	DLYPRT	RST	RSTx2	SETx2	RSTx1	SETx1	CPT2	CPT1	UPD	COT	REP	CMP4	CMP3	CMP2	CMP1
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108 (TIMA) 0x208 (TIMB) 0x308 (TIMC) 0x408 (TIMD) 0x508 (TIME) 0x608 (TIMF)	**HRTIM_ TIMxICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CORDEC	DLYPRTC	RSTC	RSTx2C	SET2xC	RST1xC	SET1xC	CPT2C	CPT1C	UPDC	COTC	REPC	CMP4C	CMP3C	CMP2C	CMP1C
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C (TIMA) 0x20C (TIMB) 0x30C (TIMC) 0x40C (TIMD) 0x50C (TIME) 0x60C (TIMF)	**HRTIM_ TIMxDIER**(1)	CORDEDE	DLYPRTDE	RSTDE	RSTx2DE	SETx2DE	RSTx1DE	SET1xDE	CPT2DE	CPT1DE	UPDDE	COTDE	REPDE	CMP4DE	CMP3DE	CMP2DE	CMP1DE	CORDEIE	DLYPRTIE	RSTIE	RSTx2IE	SETx2IE	RSTx1IE	SET1xIE	CPT2IE	CPT1IE	UPDIE	COTIE	REPIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110 (TIMA) 0x210 (TIMB) 0x310 (TIMC) 0x410 (TIMD) 0x510 (TIME) 0x610 (TIMF)	**HRTIM_ CNTxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTx[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114 (TIMA) 0x214 (TIMB) 0x314 (TIMC) 0x414 (TIMD) 0x514 (TIME) 0x614 (TIMF)	**HRTIM_ PERxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERx[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0x118 (TIMA) 0x218 (TIMB) 0x318 (TIMC) 0x418 (TIMD) 0x518 (TIME) 0x618 (TIMF)	**HRTIM_ REPxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x11C (TIMA) 0x21C (TIMB) 0x31C (TIMC) 0x41C (TIMD) 0x51C (TIME) 0x61C (TIMF)	**HRTIM_ CMP1xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP1x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x120 (TIMA) 0x220 (TIMB) 0x320 (TIMC) 0x420 (TIMD) 0x520 (TIME) 0x620 (TIMF)	**HRTIM_ CMP1CxR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPx[7:0]								CMP1x[15:0]															
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124 (TIMA) 0x224 (TIMB) 0x324 (TIMC) 0x424 (TIMD) 0x524 (TIME) 0x624 (TIMF)	**HRTIM_ CMP2xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP2x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 448. HRTIM register map and reset values – TIMx (x= A..F) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x128 (TIMA) 0x228 (TIMB) 0x328 (TIMC) 0x428 (TIMD) 0x528 (TIME) 0x628 (TIMF)	**HRTIM_CMP3xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP3x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12C (TIMA) 0x22C (TIMB) 0x32C (TIMC) 0x42C (TIMD) 0x52C (TIME) 0x62C (TIMF)	**HRTIM_CMP4xR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP4x[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x130 (TIMA) 0x230 (TIMB) 0x330 (TIMC) 0x430 (TIMD) 0x530 (TIME) 0x630 (TIMF)	**HRTIM_CPT1xR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR	CPT1x[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134 (TIMA) 0x234 (TIMB) 0x334 (TIMC) 0x434 (TIMD) 0x534 (TIME) 0x634 (TIMF)	**HRTIM_CPT2xR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR	CPT2x[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x138 (TIMA) 0x238 (TIMB) 0x338 (TIMC) 0x438 (TIMD) 0x538 (TIME) 0x638 (TIMF)	**HRTIM_DTxR**(1)	DTFLKx	DTFSLKx	Res.	Res.	DTM[1:0]		SDTFx	DTFx[8:0]									DTRLKx	DTRSLKx	Res.	DTPRSC[2:0]			SDTRx	DTRx[8:0]								
	Reset value	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x13C (TIMA) 0x23C (TIMB) 0x33C (TIMC) 0x43C (TIMD) 0x53C (TIME) 0x63C (TIMF)	**HRTIM_SETx1R**(1)	UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140 (TIMA) 0x240 (TIMB) 0x340 (TIMC) 0x440 (TIMD) 0x540 (TIME) 0x640 (TIMF)	**HRTIM_RSTx1R**(1)	UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144 (TIMA) 0x244 (TIMB) 0x344 (TIMC) 0x444 (TIMD) 0x544 (TIME) 0x644 (TIMF)	**HRTIM_SETx2R**(1)	UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x148 (TIMA) 0x248 (TIMB) 0x348 (TIMC) 0x448 (TIMD) 0x548 (TIME) 0x648 (TIMF)	**HRTIM_RSTx2R**(1)	UPDATE	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	TIMEVNT9	TIMEVNT8	TIMEVNT7	TIMEVNT6	TIMEVNT5	TIMEVNT4	TIMEVNT3	TIMEVNT2	TIMEVNT1	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP3	CMP2	CMP1	PER	RESYNC	SRT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14C (TIMA) 0x24C (TIMB) 0x34C (TIMC) 0x44C (TIMD) 0x54C (TIME) 0x64C (TIMF)	**HRTIM_EEFxR1**	Res.	Res.	EEEFLTR[4:0]					EEELTCH	EEDFLTR[4:0]					EEDLTCH	EECFLTR[4:0]					EECLTCH	EEBFLTR[4:0]					EEBLTCH	EEAFLTR[4:0]					EEALTCH
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 448. HRTIM register map and reset values – TIMx (x= A..F) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x150 (TIMA) 0x250 (TIMB) 0x350 (TIMC) 0x450 (TIMD) 0x550 (TIME) 0x650 (TIMF)	**HRTIM_ EEFxR2**	Res.	Res.	EEJFLTR[3:0]					EEJLTCH	EEIFLTR[3:0]					EEILTCH	EEHFLTR[3:0]					EEHLTCH	EEGFLTR[3:0]					EEGLTCH	EEFFLTR[3:0]					EEFLTCH
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	**HRTIM_ RSTAR(1)**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x254	**HRTIM_ RSTBR(1)**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x354	**HRTIM_ RSTCR(1)**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x454	**HRTIM_ RSTDR(1)**	TIMFCMP2	TIMECMP4	TIMECMP2	TIMECMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x554	**HRTIM_ RSTER(1)**	TIMFCMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMFCMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x654	**HRTIM_ RSTFR(1)**	TIMECMP2	TIMDCMP4	TIMDCMP2	TIMDCMP1	TIMCCMP4	TIMCCMP2	TIMCCMP1	TIMBCMP4	TIMBCMP2	TIMBCMP1	TIMACMP4	TIMACMP2	TIMACMP1	EXTEVNTJ	EXTEVNTI	EXTEVNTH	EXTEVNTG	EXTEVNTF	EXTEVNTE	EXTEVNTD	EXTEVNTC	EXTEVNTB	EXTEVNTA	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTPER	CMP4	CMP2	UPDT	TIMECMP1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x158 (TIMA) 0x258 (TIMB) 0x358 (TIMC) 0x458 (TIMD) 0x558 (TIME) 0x658 (TIMF)	**HRTIM_ CHPxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRTPW [3:0]					CARDT Y[2:0]			CARFRQ [3:0]			
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x15C	**HRTIM_ CPT1ACR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x25C	**HRTIM_ CPT1BCR**	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TA1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 448. HRTIM register map and reset values – TIMx (x= A..F) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x35C	HRTIM_ CPT1CCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x45C	HRTIM_ CPT1DCR	TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x55C	HRTIM_ CPT1ECR	TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x65C	HRTIM_ CPT1FCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x160	HRTIM_ CPT2ACR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x260	HRTIM_ CPT2BCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x360	HRTIM_ CPT2CCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x460	HRTIM_ CPT2DCR	TECMP2	TECMP1	TE1RST	TE1SET	TFCMP2	TFCMP1	TF1RST	TF1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	HRTIM_ CPT2ECR	TFCMP2	TFCMP1	TF1RST	TF1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x660	HRTIM_ CPT2FCR	TECMP2	TECMP1	TE1RST	TE1SET	TDCMP2	TDCMP1	TD1RST	TD1SET	TCCMP2	TCCMP1	TC1RST	TC1SET	TBCMP2	TBCMP1	TB1RST	TB1SET	TACMP2	TACMP1	TA1RST	TA1SET	EXEVJCPT	EXEVICPT	EXEVHCPT	EXEVGCP	EXEVFCPT	EXEVECPT	EXEVDCPT	EXEVCCPT	EXEVBCPT	EXEVACPT	UPDCPT	SWCPT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 448. HRTIM register map and reset values – TIMx (x= A..F) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x164 (TIMA) 0x264 (TIMB) 0x364 (TIMC) 0x464 (TIMD) 0x564 (TIME) 0x664 (TIMF)	**HRTIM_ OUTxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIDL2	CHP2	FAULT2[1:0]		IDLES2	IDLEM2	POL2	Res.	Res.	Res.	Res.	DLYPRT[2:0]		DLYPRTEN	DTEN	DIDL1	CHP1	FAULT1[1:0]		IDLES1	IDLEM1	POL1	Res.
	Reset value									0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	
0x168 (TIMA) 0x268 (TIMB) 0x368 (TIMC) 0x468 (TIMD) 0x568 (TIME) 0x668 (TIMF)	**HRTIM_ FLTxR**	FLTLCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT7A8EN	FLT6A7EN	FLT5A6EN	FLT4A5EN	FLT3A4EN	FLT2A3EN	FLT1A2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8EN	FLT7EN	FLT6EN	FLT5EN	FLT4EN	FLT3EN	FLT2EN	FLT1EN
	Reset value	0									0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x16C (TIMA) 0x26C(TIMB) 0x36C (TIMC) 0x46C (TIMD) 0x56C (TIME) 0x66C (TIMF)	**HRTIM_ TIMxCR2**	Res.	Res.	Res.	Res.	DCDCOE	COM	COR	COE	ADCTRGCH2	ADCTRGCH1	TRGHLFT	TRGHLF	Res.	Res.	GTCMP3	GTCMP1	FE ROM [1:0]		BM ROM [1:0]		AD ROM [1:0]		OUT ROM [1:0]		ROM [1:0]		Res.	UDM	Res.	DCDR	DCDS	DCDE
	Reset value					0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0
0x170 (TIMA) 0x270 (TIMB) 0x370 (TIMC) 0x470 (TIMD) 0x570 (TIME) 0x670 (TIMF)	**HRTIM_ TIMxEEFR3**	Res.	Res.	EEVZCNT[5:0]						EEVZ CNT[5:0]				Res.	EEVZRSTM	EEVZCRES	EEVZCE	Res.	Res.	EEVYCNT[5:0]						EEVYSEL [5:0]				Res.	EEVYRSTM	EEVYCRES	EEVYCE
	Reset value			0	0	0	0	0	0	0	0	0	0		0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0
0x174 (TIMA) 0x274 (TIMB) 0x374 (TIMC) 0x474 (TIMD) 0x574 (TIME) 0x674 (TIMF)	**HRTIM_ EESELxR**	Res.	Res.	EEVJSEL[1:0]		Res.	EEVISEL[1:0]		Res.	EEVHSEL[1:0]		Res.	EEVGSEL[1:0]	Res.	Res.	EEVFSEL[1:0]		Res.	Res.	EEVESEL[1:0]		Res.	EEVDSEL[1:0]		Res.	EEVCSEL[1:0]		Res.	EEVBSEL[1:0]		Res.	EEVASEL[1:0]	
	Reset value			0	0		0	0		0	0		0	0		0	0			0	0		0	0		0	0		0	0		0	0	0
0x178 (TIMA) 0x278 (TIMB) 0x378 (TIMC) 0x478 (TIMD) 0x578 (TIME) 0x678 (TIMF)	**HRTIM_ COTRGxR**	EXT_TRG5	EXT_TRG4	EXT_TRG3	EXT_TRG2	EXT_TRG1	EXT_TRG0	EEVJ	EEVI	EEVH	EEVG	EEVF	EEVE	EEVD	EEVC	EEVB	EEVA	FLT7A8	FLT6A7	FLT5A6	FLT4A5	FLT3A4	FLT2A3	FLT1A2	FLT8	FLT7	FLT6	FLT5	FLT4	FLT3	FLT2	FLT1	SWT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17C (TIMA) 0x27C (TIMB) 0x37C (TIMC) 0x47C (TIMD) 0x57C (TIME) 0x67C (TIMF)	**HRTIM_ COFExR**	Res.	Res.	EEVJFRCB	EEVIFRCB	EEVHFRCB	EEVGFRCB	EEVFFRCB	EEVEFRCB	EEVDFRCB	EEVCFRCB	EEVBFRCB	EEVAFRCB	Res.	COFMB[2:0]			Res.	Res.	EEVJFRCA	EEVIFRCA	EEVHFRCA	EEVGFRCA	EEVFFRCA	EEVEFRCA	EEVDFRCA	EEVCFRCA	EEVBFRCA	EEVAFRCA	Res.	COFMA[2:0]		
	Reset value			0	0	0	0	0	0	0	0	0	0		0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0
0x180 (TIMA) 0x280 (TIMB) 0x380 (TIMC) 0x480 (TIMD) 0x580 (TIME) 0x680 (TIMF)	**HRTIM_ CODFxR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLY2FLTD[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLY1FLTD[8:0]								
	Reset value								0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0

1. This register can be preloaded (see *Table 433 on page 1660*).

Table 449. HRTIM register map and reset values – common functions

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x700	**HRTIM_CR1**	Res.	Res.	Res.	Res.	AD4USRC[2:0]			AD3USRC[2:0]			AD2USRC[2:0]			AD1USRC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFUDIS	TEUDIS	TDUDIS	TCUDIS	TBUDIS	TAUDIS	MUDIS
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x704	**HRTIM_CR2**	Res.	Res.	Res.	Res.	Res.	DBSS [2:0]			Res.	Res.	SWPF	SWPE	SWPD	SWPC	SWPB	SWPA	Res.	TFRST	TERST	TDRST	TCRST	TBRST	TARST	MRST	Res.	TFSWU	TESWU	TDSWU	TCSWU	TBSWU	TASWU	MSWU
	Reset value						0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x708	**HRTIM_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER	DLLRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8	FLT7	FLT6	SYSFLT	FLT5	FLT4	FLT3	FLT2	FLT1
	Reset value															0	0								0	0	0	0	0	0	0	0	0
0x70C	**HRTIM_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERC	DLLRDYC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8C	FLT7C	FLT6C	SYSFLTC	FLT5C	FLT4C	FLT3C	FLT2C	FLT1C
	Reset value															0	0								0	0	0	0	0	0	0	0	0
0x710	**HRTIM_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPERIE	DLLRDYIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8IE	FLT7IE	FLT6IE	SYSFLTIE	FLT5IE	FLT4IE	FLT3IE	FLT2IE	FLT1IE
	Reset value															0	0								0	0	0	0	0	0	0	0	0
0x714	**HRTIM_OENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2OEN	TF1OEN	TE2OEN	TE1OEN	TD2OEN	TD1OEN	TC2OEN	TC1OEN	TB2OEN	TB1OEN	TA2OEN	TA1OEN
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x718	**HRTIM_ODISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2ODIS	TF1ODIS	TE2ODIS	TE1ODIS	TD2ODIS	TD1ODIS	TC2ODIS	TC1ODIS	TB2ODIS	TB1ODIS	TA2ODIS	TA1ODIS
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x71C	**HRTIM_ODSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TF2ODS	TF1ODS	TE2ODS	TE1ODS	TD2ODS	TD1ODS	TC2ODS	TC1ODS	TB2ODS	TB1ODS	TA2ODS	TA1ODS
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x720	**HRTIM_BMCR**	BMSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFBM	TEBM	TDBM	TCBM	TBBM	TABM	MTBM	Res.	Res.	Res.	Res.	Res.	BMPREN	BMPRSC[3:0]				BMCLK[3:0]				BMOM	BME
	Reset value	0									0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x724	**HRTIM_ BMTRGR**	OCHPEV	EEV8	EEV7	TDEEV8	TAEEV7	TECMP2	TECMP1	TEREP	TFCMP1	TDCMP2	TFREP	TDREP	TDRST	TFRST	TCCMP1	TCREP	TCRST	TBCMP2	TBCMP1	TBREP	TBRST	TACMP2	TACMP1	TAREP	TARST	MSTCMP4	MSTCMP3	MSTCMP2	MSTCMP1	MSTREP	MSTRST	SW
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x728	**HRTIM_ BMCMPR**(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMCMP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 449. HRTIM register map and reset values – common functions (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x72C	HRTIM_BMPER(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BMPER[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x730	HRTIM_EECR1	Res.	Res.	EE5FAST	EE5SNS[1:0]		EE5POL	EE5SRC[1:0]		EE4FAST	EE4SNS[1:0]		EE4POL	EE4SRC[1:0]		EE3FAST	EE3SNS[1:0]		EE3POL	EE3SRC[1:0]		EE2FAST	EE2SNS[1:0]		EE2POL	EE2SRC[1:0]		EE1FAST	EE1SNS[1:0]		EE1POL	EE1SRC[1:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x734	HRTIM_EECR2	Res.	Res.	Res.	EE10SNS[1:0]		EE10POL	EE10SRC[1:0]		Res.	EE9SNS[1:0]		EE9POL	EE9SRC[1:0]		Res.	EE8SNS[1:0]		EE8POL	EE8SRC[1:0]		Res.	EE7SNS[1:0]		EE7POL	EE7SRC[1:0]		Res.	EE6SNS[1:0]		EE6POL	EE6SRC[1:0]	
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x738	HRTIM_EECR3	EEVSD[1:0]		Res.	Res.	EE10F[3:0]				Res.	Res.	EE9F[3:0]				Res.	Res.	EE8F[3:0]				Res.	Res.	EE7F[3:0]				Res.	Res.	EE6F[3:0]			
	Reset value	0	0			0	0	0	0			0	0	0	0			0	0	0	0			0	0	0	0			0	0	0	0
0x73C	HRTIM_ADC1R(1)	ADC1TEPER	ADC1TEC4	ADC1TEC3	ADC1TFRST	ADC1TDPER	ADC1TDC4	ADC1TDC3	ADC1TFPER	ADC1TCPER	ADC1TCC4	ADC1TCC3	ADC1TFC4	ADC1TBRST	ADC1TBPER	ADC1TBC4	ADC1TBC3	ADC1TFC3	ADC1TARST	ADC1TAPER	ADC1TAC4	ADC1TAC3	ADC1TFC2	ADC1EEV5	ADC1EEV4	ADC1EEV3	ADC1EEV2	ADC1EEV1	ADC1MPER	ADC1MC4	ADC1MC3	ADC1MC2	ADC1MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x740	HRTIM_ADC2R(1)	ADC2TERST	ADC2TEC4	ADC2TEC3	ADC2TEC2	ADC2TDRST	ADC2TDPER	ADC2TDC4	ADC2TFPER	ADC2TDC2	ADC2TCRST	ADC2TCPER	ADC2TCC4	ADC2TFC4	ADC2TCC2	ADC2TBPER	ADC2TBC4	ADC2TFC3	ADC2TBC2	ADC2TAPER	ADC2TAC4	ADC2TAC3	ADC2TFC2	ADC2TAC2	ADC2EEV10	ADC2EEV9	ADC2EEV8	ADC2EEV7	ADC2EEV6	ADC2MPER	ADC2MC4	ADC2MC3	ADC2MC2	ADC2MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x744	HRTIM_ADC3R(1)	ADC3TEPER	ADC3TEC4	ADC3TEC3	ADC3TFRST	ADC3TDPER	ADC3TDC4	ADC3TDC3	ADC3TFPER	ADC3TCPER	ADC3TCC4	ADC3TCC3	ADC3TFC4	ADC3TBRST	ADC3TBPER	ADC3TBC4	ADC3TBC3	ADC3TFC3	ADC3TARST	ADC3TAPER	ADC3TAC4	ADC3TAC3	ADC3TFC2	ADC3EEV5	ADC3EEV4	ADC3EEV3	ADC3EEV2	ADC3EEV1	ADC3MPER	ADC3MC4	ADC3MC3	ADC3MC2	ADC3MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x748	HRTIM_ADC4R(1)	ADC4TERST	ADC4TEC4	ADC4TEC3	ADC4TEC2	ADC4TDRST	ADC4TDPER	ADC4TDC4	ADC4TFPER	ADC4TDC2	ADC4TCRST	ADC4TCPER	ADC4TCC4	ADC4TFC4	ADC4TCC2	ADC4TBPER	ADC4TBC4	ADC4TFC3	ADC4TBC2	ADC4TAPER	ADC4TAC4	ADC4TAC3	ADC4TFC2	ADC4TAC2	ADC4EEV10	ADC4EEV9	ADC4EEV8	ADC4EEV7	ADC4EEV6	ADC4MPER	ADC4MC4	ADC4MC3	ADC4MC2	ADC4MC1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74C	HRTIM_DLLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALRTE[1:0]		CALEN	CAL
	Reset value																													0	0	0	0

Table 449. HRTIM register map and reset values – common functions (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x750	**HRTIM_ FLTINR1**	FLT4LCK	FLT4F[3:0]				FLT4SRC[0]	FLT4P	FLT4E	FLT3LCK	FLT3F[3:0]				FLT3SRC[0]	FLT3P	FLT3E	FLT2LCK	FLT2F[3:0]				FLT2SRC[0]	FLT2P	FLT2E	FLT1LCK	FLT1F[3:0]				FLT1SRC[0]	FLT1P	FLT1E
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x754	**HRTIM_ FLTINR2**	Res.	Res.	Res.	Res.	Res.	Res.	FLTSD[1:0]		FLT8SRC[1]	FLT7SRC[1]	FLT6SRC[1]	FLT5SRC[1]	FLT4SRC[1]	FLT3SRC[1]	FLT2SRC[1]	FLT1SRC[1]	FLT6LCK	FLT6F[3:0]				FLT6SRC[0]	FLT6P	FLT6E	FLT5LCK	FLT5F[3:0]				FLT5SRC[0]	FLT5P	FLT5E
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x758	**HRTIM_ BDMUPDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCMP4	MCMP3	MCMP2	MCMP1	MREP	MPER	MCNT	MDIER	MICR	MCR
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x75C	**HRTIM_ BDTAUPR**	Res.	Res.	Res.	Res.	Res.	TIMACODER	TIMACOFER	TIMACOTRGR	TIMAEESELR	TIMAEEFR3	TIMACR2	TIMAFLTR	TIMAOUTR	TIMACHPR	TIMARSTR	TIMAEEFR2	TIMAEEFR1	TIMARST2R	TIMASET2R	TIMARST1R	TIMASET1R	TIMADTxR	TIMACMP4	TIMACMP3	TIMACMP2	TIMACMP1	TIMAREP	TIMAPER	TIMACNT	TIMADIER	TIMAICR	TIMACR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x760	**HRTIM_ BDTBUPR**	Res.	Res.	Res.	Res.	Res.	TIMBCODER	TIMBCOFER	TIMBCOTRGR	TIMBEESELR	TIMBEEFR3	TIMBCR2	TIMBFLTR	TIMBOUTR	TIMBCHPR	TIMBRSTR	TIMBEEFR2	TIMBEEFR1	TIMBRST2R	TIMBSET2R	TIMBRST1R	TIMBSET1R	TIMBDTxR	TIMBCMP4	TIMBCMP3	TIMBCMP2	TIMBCMP1	TIMBREP	TIMBPER	TIMBCNT	TIMBDIER	TIMBICR	TIMBCR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x764	**HRTIM_ BDTCUPR**	Res.	Res.	Res.	Res.	Res.	TIMCCODER	TIMCCOFER	TIMCCOTRGR	TIMCEESELR	TIMCEEFR3	TIMCCR2	TIMCFLTR	TIMCOUTR	TIMCCHPR	TIMCRSTR	TIMCEEFR2	TIMCEEFR1	TIMCRST2R	TIMCSET2R	TIMCRST1R	TIMCSET1R	TIMCDTxR	TIMCCMP4	TIMCCMP3	TIMCCMP2	TIMCCMP1	TIMCREP	TIMCPER	TIMCCNT	TIMCDIER	TIMCICR	TIMCCR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x768	**HRTIM_ BDTDUPR**	Res.	Res.	Res.	Res.	Res.	TIMDCODER	TIMDCOFER	TIMDCOTRGR	TIMDEESELR	TIMDEEFR3	TIMDCR2	TIMDFLTR	TIMDOUTR	TIMDCHPR	TIMDRSTR	TIMDEEFR2	TIMDEEFR1	TIMDRST2R	TIMDSET2R	TIMDRST1R	TIMDSET1R	TIMDDTxR	TIMDCMP4	TIMDCMP3	TIMDCMP2	TIMDCMP1	TIMDREP	TIMDPER	TIMDCNT	TIMDDIER	TIMDICR	TIMDCR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x76C	**HRTIM_ BDTEUPR**	Res.	Res.	Res.	Res.	Res.	TIMECODER	TIMECOFER	TIMECOTRGR	TIMEEESELR	TIMEEEFR3	TIMECR2	TIMEFLTR	TIMEOUTR	TIMECHPR	TIMERSTR	TIMEEEFR2	TIMEEEFR1	TIMERST2R	TIMESET2R	TIMERST1R	TIMESET1R	TIMEDTxR	TIMECMP4	TIMECMP3	TIMECMP2	TIMECMP1	TIMEREP	TIMEPER	TIMECNT	TIMEDIER	TIMEICR	TIMECR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x770	**HRTIM_ BDMADR**	BDMADR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 449. HRTIM register map and reset values – common functions (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x774	**HRTIM_ BDTFUPR**	Res.	Res.	Res.	Res.	Res.	TIMFCODER	TIMFCOFER	TIMFCOTRGR	TIMFEESELR	TIMFEEFR3	TIMFCR2	TIMFFLTR	TIMFOUTR	TIMFCHPR	TIMFRSTR	TIMFEEFR2	TIMFEEFR1	TIMFRST2R	TIMFSET2R	TIMFRST1R	TIMFSET1R	TIMFDTxR	TIMFCMP4	TIMFCMP3	TIMFCMP2	TIMFCMP1	TIMFREP	TIMFPER	TIMFCNT	TIMFDIER	TIMFICR	TIMFCR
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x778	**HRTIM_ ADCER**(1)	Res.	ADC10TRG[4:0]					ADC9TRG[4:0]					ADC8TRG[4:0]					Res.	ADC7TRG[4:0]					ADC6TRG[4:0]					ADC5TRG[4:0]				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x77C	**HRTIM_ ADCUR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AD10U SRC [2:0]			Res.	AD9U SRC [2:0]			Res.	AD8U SRC [2:0]			Res.	AD7U SRC [2:0]			Res.	AD6U SRC [2:0]			Res.	AD5U SRC [2:0]		
	Reset value										0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x780	**HRTIM_ADCPS1**	Res.	Res.	Res.	ADC5PSC[4:0]					Res.	ADC4PSC[4:0]					Res.	ADC3PSC[4:0]					Res.	ADC2PSC[4:0]					Res.	ADC1PSC[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x784	**HRTIM_ADCPS2**	Res.	Res.	Res.	ADC10PSC[4:0]					Res.	ADC9PSC[4:0]					Res.	ADC8PSC[4:0]					Res.	ADC7PSC[4:0]					Res.	ADC6PSC[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x788	**HRTIM_ FLTINR3**	FLT4RSTM	FLT4CRES	FLT4CNT[3:0]				FLT4BLKS	FLT4BLKE	FLT3RSTM	FLT3CRES	FLT3CNT[3:0]				FLT3BLKS	FLT3BLKE	FLT2RSTM	FLT2CRES	FLT2CNT[3:0]				FLT2BLKS	FLT2BLKE	FLT1RSTM	FLT1CRES	FLT1CNT[3:0]				FLT1BLKS	FLT1BLKE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78C	**HRTIM_ FLTINR4**	FLT8RSTM	FLT8CRES	FLT8CNT[3:0]				FLT8BLKS	FLT8BLKE	FLT7RSTM	FLT7CRES	FLT7CNT[3:0]				FLT7BLKS	FLT7BLKE	FLT6RSTM	FLT6CRES	FLT6CNT[3:0]				FLT6BLKS	FLT6BLKE	FLT5RSTM	FLT5CRES	FLT5CNT[3:0]				FLT5BLKS	FLT5BLKE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x790	HRTIM_ FLTINR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLT8LCK	FLT8F[3:0]				FLT8SRC[0]	FLT8P	FLT8E	FLT7LCK	FLT7F[3:0]				FLT7SRC[0]	FLT7P	FLT7E
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x794-0x79C	Reserved	Res.																															
0x7A0	HRTIM_EECR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE13FAST	EE13SNS[1:0]		EE13POL	EE13SRC[1:0]		EE12FAST	EE12SNS[1:0]		EE12POL	EE12SRC[1:0]		EE11FAST	EE11SNS[1:0]		EE11POL	EE11SRC[1:0]	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7A4	HRTIM_EECR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE16SNS[1:0]		EE16POL	EE16SRC[1:0]		Res.	EE15SNS[1:0]		EE15POL	EE15SRC[1:0]		Res.	EE14SNS[1:0]		EE14POL	EE14SRC[1:0]	
	Reset value																0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0

Table 449. HRTIM register map and reset values – common functions (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7A8	HRTIM_EECR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EE16F[3:0]				Res.	Res.	EE15F[3:0]				Res.	Res.	EE14F[3:0]			
	Reset value																	0	0	0	0			0	0	0	0			0	0	0	0
0x7AC	HRTIM_EECR7	Res.	EE16SRC[2]	Res.	EE15SRC[2]	Res.	EE14SRC[2]	Res.	EE13SRC[2]	Res.	EE12SRC[2]	Res.	EE11SRC[2]	Res.	EE10SRC[2]	Res.	EE9SRC[2]	Res.	EE8SRC[2]	Res.	EE7SRC[2]	Res.	EE6SRC[2]	Res.	EE5SRC[2]	Res.	EE4SRC[2]	Res.	EE3SRC[2]	Res.	EE2SRC[2]	Res.	EE1SRC[2]
	Reset value		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0		0
0x7B0-0x7BC	Reserved	Res.																															
0x7C0	HRTIM_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTFUDISA	RTEUDISA	RTDUDISA	RTCUDISA	RTBUDISA	RTAUDISA	RMUDISA
	Reset value																										0	0	0	0	0	0	0
0x7FC	HRTIM_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSYNCSEL	UPDENSEL	BMTRGSEL
	Reset value																														0	0	0

1. This register can be preloaded (see *Table 433 on page 1660*).

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

41 Advanced-control timers (TIM1/TIM8/TIM20)

41.1 TIM1/TIM8/TIM20 introduction

The advanced-control timers (TIM1/TIM8/TIM20) consist of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8/TIM20) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 41.3.35: Timer synchronization*.

41.2 TIM1/TIM8/TIM20 main features

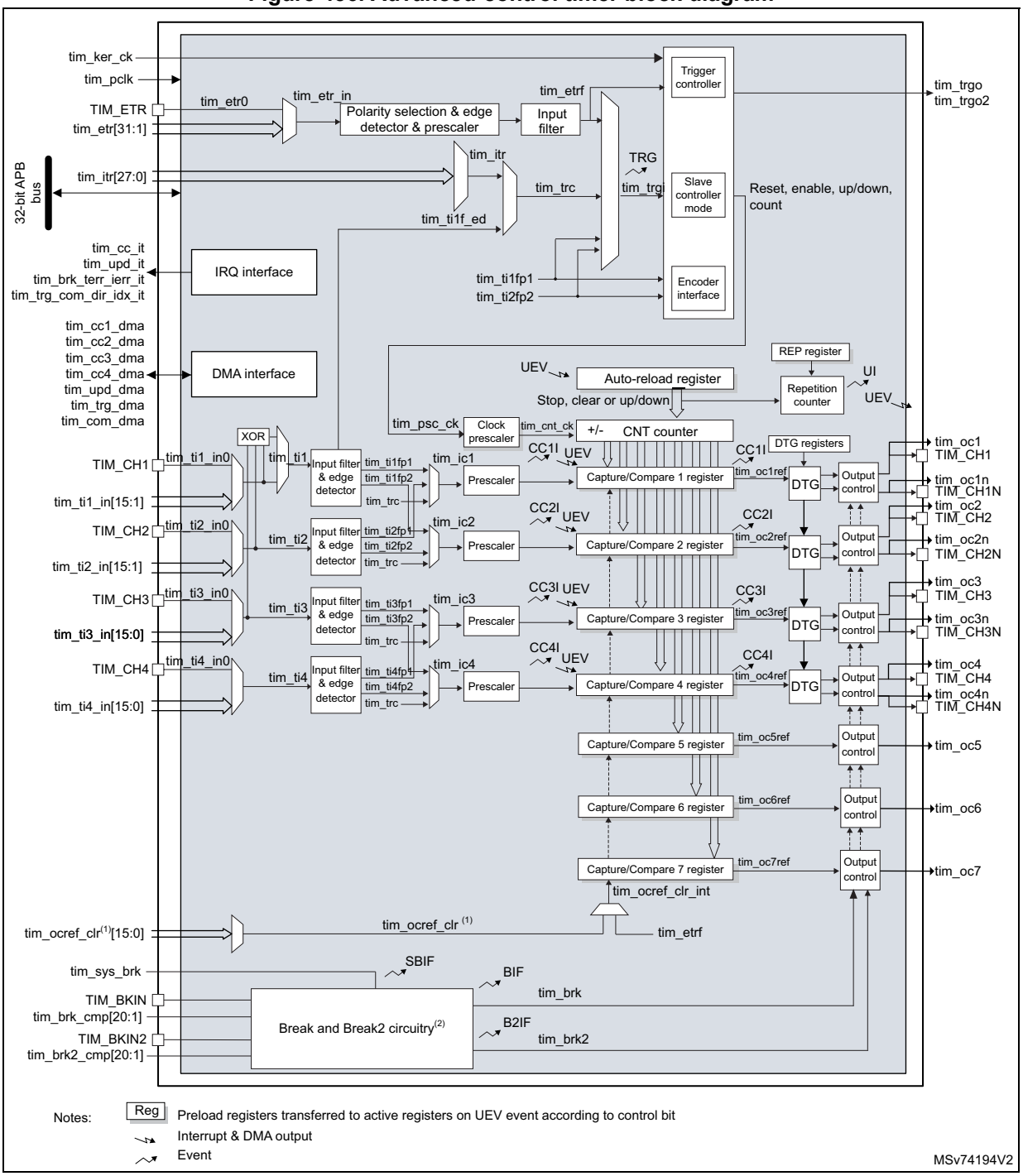
TIM1/TIM8/TIM20 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Seven independent channels for:
  - Input capture (but channels 5, 6 and 7)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
- Incremental encoders, quadrature encoders and hall-sensors support
- Trigger input for external clock or cycle-by-cycle current management
- ADC synchronization for jitter-free sampling points

41.3 TIM1/TIM8/TIM20 functional description

41.3.1 Block diagram

Figure 456. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals*.
2. See *Figure 507: Break and Break2 circuitry overview* for details.

41.3.2 TIM1/TIM8/TIM20 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 450. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input / Output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 451. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[31:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 451. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocx (x = 1 to 4)	Output	Internal output compare signals of channels 1 to 4. These outputs can be used for triggering/synchronizing other internal peripherals.
tim_ocxn (x = 1 to 4)	Output	Internal timer complementary outputs, derived from tim_ocx internal outputs with the possibility to have deadtime insertion.
tim_ocx (x = 5 to 7)	Output	Internal output compare signals of channels 5 to 7. These outputs can be used for triggering/synchronizing other internal peripherals.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[20:1]	Input	Break input for internal signals
tim_brk2_cmp[20:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trgi_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 452, *Table 453*, *Table 454*, and *Table 455* list the sources connected to the tim_ti[4:1] input multiplexers.

Table 452. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti1_in0	TIM1_CH1	TIM8_CH1	TIM20_CH1
tim_ti1_in1	comp1_out		
tim_ti1_in2	comp2_out		
tim_ti1_in3	comp3_out		
tim_ti1_in4	comp4_out		
tim_ti1_in[5:8]	Reserved		
tim_ti1_in9	play1_out0(1)	play1_out14(1)	play1_out3(1)
tim_ti1_in[15:10]	Reserved		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 453. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti2_in0	TIM1_CH2	TIM8_CH2	TIM20_CH2
tim_ti2_in[8:1]	Reserved		
tim_ti2_in9	play1_out2(1)	play1_out10(1)	play1_out7(1)
tim_ti2_in[15:10]	Reserved		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 454. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti3_in0	TIM1_CH3	TIM8_CH3	TIM20_CH3
tim_ti3_in[2:1]	Reserved		
tim_ti3_in3	play1_out1(1)	play1_out12(1)	play1_out8(1)
tim_ti3_in[15:4]	Reserved		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 455. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti4_in0	TIM1_CH4	TIM8_CH4	TIM20_CH4
tim_ti4_in[2:1]	Reserved		

Table 455. Interconnect to the tim_ti4 input multiplexer (continued)

tim_ti4 inputs	Sources		
	TIM1	TIM8	TIM20
tim_ti4_in3	play1_out8(1)	play1_out5(1)	play1_out9(1)
tim_ti4_in[15:4]	Reserved		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 456 lists the internal sources connected to the tim_itr input multiplexer.

Table 456. Internal trigger connection

Timer internal trigger input signal	TIM1	TIM8	TIM20
tim_itr0	Reserved	tim1_trgo	
tim_itr1		tim1_trgo2	
tim_itr2	tim2_trgo		
tim_itr3	tim3_trgo(1)		
tim_itr4	tim4_trgo(1)		
tim_itr5	tim5_trgo(1)		
tim_itr6	tim8_trgo	Reserved	tim8_trgo
tim_itr7	tim8_trgo2		tim8_trgo2
tim_itr8	tim9_trgo		
tim_itr9	tim10_oc1		
tim_itr10	tim11_oc1		
tim_itr11	tim12_trgo		
tim_itr12	tim13_oc1		
tim_itr13	tim14_oc1		
tim_itr14	tim15_trgo		
tim_itr15	tim16_oc1		
tim_itr16	tim17_oc1		
tim_itr17	tim20_trgo		Reserved
tim_itr18	tim20_trgo2		
tim_itr19	Reserved		
tim_itr20	Reserved		
tim_itr21	hrtim_out_sync2		
tim_itr[27:22]	Reserved		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 457 lists the internal sources connected to the tim_etr input multiplexer.

Table 457. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment		
	TIM1	TIM8	TIM20
tim_etr0	TIM1_ETR	TIM8_ETR	TIM20_ETR
tim_etr1	comp1_out		
tim_etr2	comp2_out		
tim_etr3	comp3_out		
tim_etr4	comp4_out		
tim_etr5	comp5_out		
tim_etr6	comp6_out		
tim_etr7	comp7_out		
tim_etr8	comp8_out		
tim_etr[16:9]	Reserved		
tim_etr17	adc1_awd1	adc2_awd1	adc3_awd1
tim_etr18	adc1_awd2	adc2_awd2	adc3_awd2
tim_etr19	adc1_awd3	adc2_awd3	adc3_awd3
tim_etr20	Reserved	adc3_awd1	Reserved
tim_etr21		adc3_awd2	
tim_etr22		adc3_awd3	
tim_etr23	play1_out6(1)		
tim_etr24	play1_out8(1)	play1_out10(1)	play1_out14(1)
tim_etr[31:25]	Reserved		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 458, *Table 459*, and *Table 461* list the sources connected to the tim_brk and tim_brk2 inputs.

Table 458. Interconnect to the tim_brk_cmp multiplexer

tim_brk inputs	TIM1	TIM8	TIM20
TIM_BKIN	TIM1_BKIN pin	TIM8_BKIN pin	TIM20_BKIN pin
tim_brk_cmp1	comp1_out		
tim_brk_cmp2	comp2_out		
tim_brk_cmp3	comp3_out		
tim_brk_cmp4	comp4_out		
tim_brk_cmp5	comp5_out		
tim_brk_cmp6	comp6_out		
tim_brk_cmp7	comp7_out		

Table 458. Interconnect to the tim_brk_cmp multiplexer

tim_brk inputs	TIM1	TIM8	TIM20
tim_brk_cmp8	comp8_out		
tim_brk_cmp9	HRTIM1_FLT1		
tim_brk_cmp10	HRTIM1_FLT2		
tim_brk_cmp[12:11]	Reserved		
tim_brk_cmp13	play1_out0(1)		
tim_brk_cmp14	play1_out1(1)	play1_out13(1)	play1_out9(1)
tim_brk_cmp15	Reserved	TIM1_BKIN	
tim_brk_cmp16	TIM8_BKIN	Reserved	TIM8_BKIN
tim_brk_cmp17	TIM20_BKIN		Reserved
tim_brk_cmp18	TIM15_BKIN		
tim_brk_cmp19	TIM16_BKIN		
tim_brk_cmp20	TIM17_BKIN		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 459. Interconnect to the tim_brk2_cmp multiplexer

tim_brk2 inputs	TIM1	TIM8	TIM20
TIM_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin	TIM20_BKIN2 pin
tim_brk2_cmp1	comp1_out		
tim_brk2_cmp2	comp2_out		
tim_brk2_cmp3	comp3_out		
tim_brk2_cmp4	comp4_out		
tim_brk2_cmp5	comp5_out		
tim_brk2_cmp6	comp6_out		
tim_brk2_cmp7	comp7_out		
tim_brk2_cmp8	comp8_out		
tim_brk2_cmp[12:9]	Reserved		
tim_brk2_cmp13	play1_out2(1)		
tim_brk2_cmp14	play1_out3(1)	play1_out15(1)	play1_out11(1)
tim_brk2_cmp15	Reserved	TIM1_BKIN2	
tim_brk2_cmp16	TIM8_BKIN2	Reserved	TIM8_BKIN2
tim_brk2_cmp17	TIM20_BKIN2		Reserved
tim_brk2_cmp[20:18]	Reserved		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 460. Interconnect to tim_ocref_clr multiplexer

tim_brk2 inputs	TIM1	TIM8	TIM20
tim_ocref_clr0	comp1_out		
tim_ocref_clr1	comp2_out		
tim_ocref_clr2	comp3_out		
tim_ocref_clr3	comp4_out		
tim_ocref_clr4	comp5_out		
tim_ocref_clr5	comp6_out		
tim_ocref_clr6	comp7_out		
tim_ocref_clr7	comp8_out		
tim_ocref_clr[11:8]	Reserved		
tim_ocref_clr12	play1_out0(1)	play1_out5(1)	play1_out12(1)
tim_ocref_clr13	play1_out6(1)	play1_out14(1)	play1_out1(1)
tim_ocref_clr[15:14]	Reserved		

1. Not available on all STM32G5 devices. Refer to the device datasheet for availability.

Table 461. System break interconnect

tim_sys_brk inputs	TIM1 / TIM8 / TIM20	Enable bit in SBS_CFGR2 register
tim_sys_brk0	Cortex-M33 LOCKUP	CLL
tim_sys_brk1	SRAM double ECC error	SEL
tim_sys_brk2	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk3	Non-Volatile memory double ECC error	ECCL
tim_sys_brk4	Cortex-M33 Data Cache ECC error	DCECCL
tim_sys_brk5	Clock Security System (CSS)	None (always enabled)

41.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)
- Repetition counter register (TIM_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 457 and *Figure 458* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 457. Counter timing diagram with prescaler division change from 1 to 2

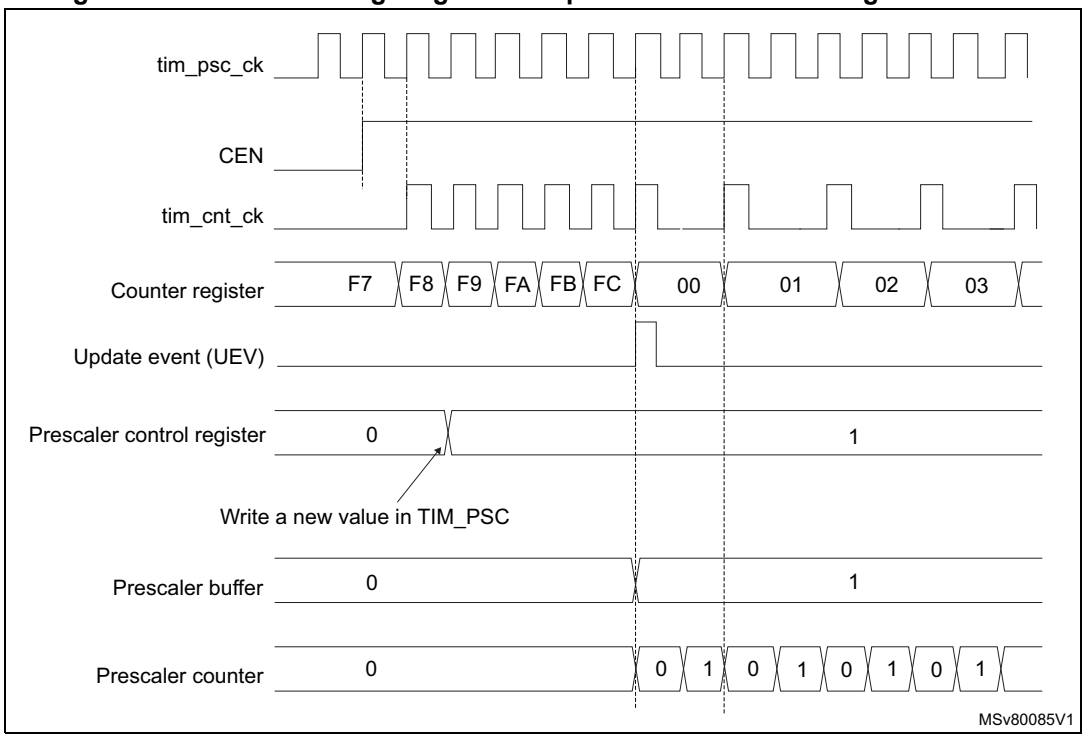
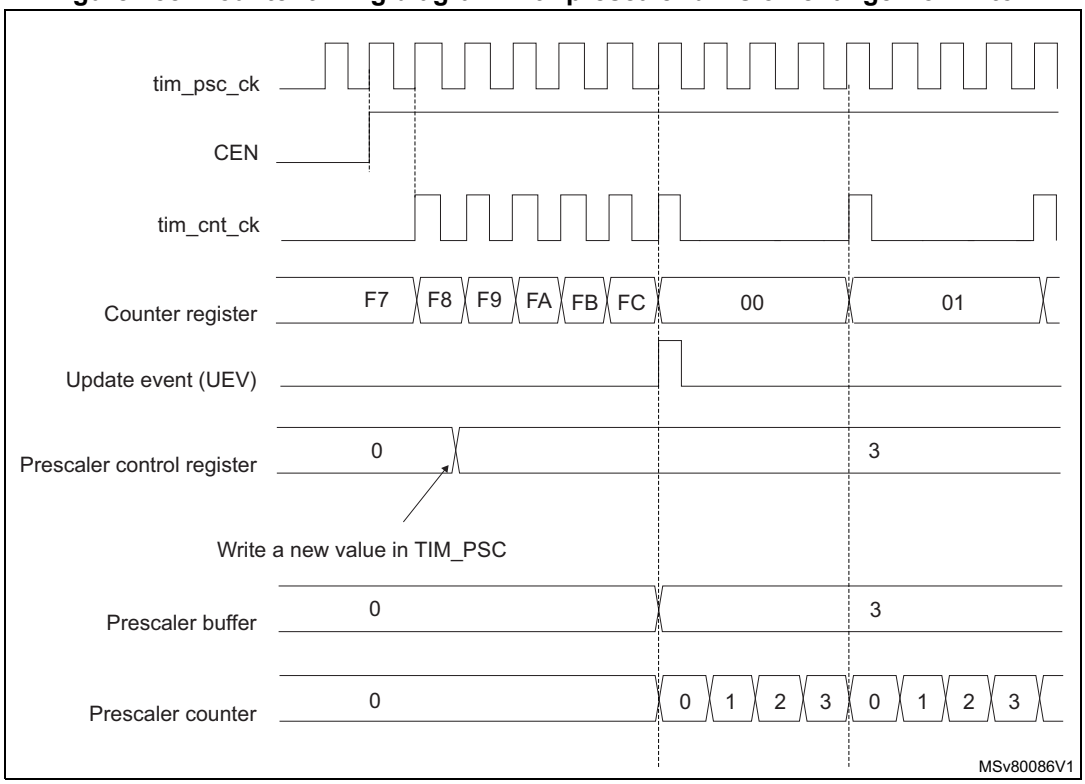


Figure 458. Counter timing diagram with prescaler division change from 1 to 4



41.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register,
- The autoreload shadow register is updated with the preload value (TIM_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 459. Counter timing diagram, internal clock divided by 1

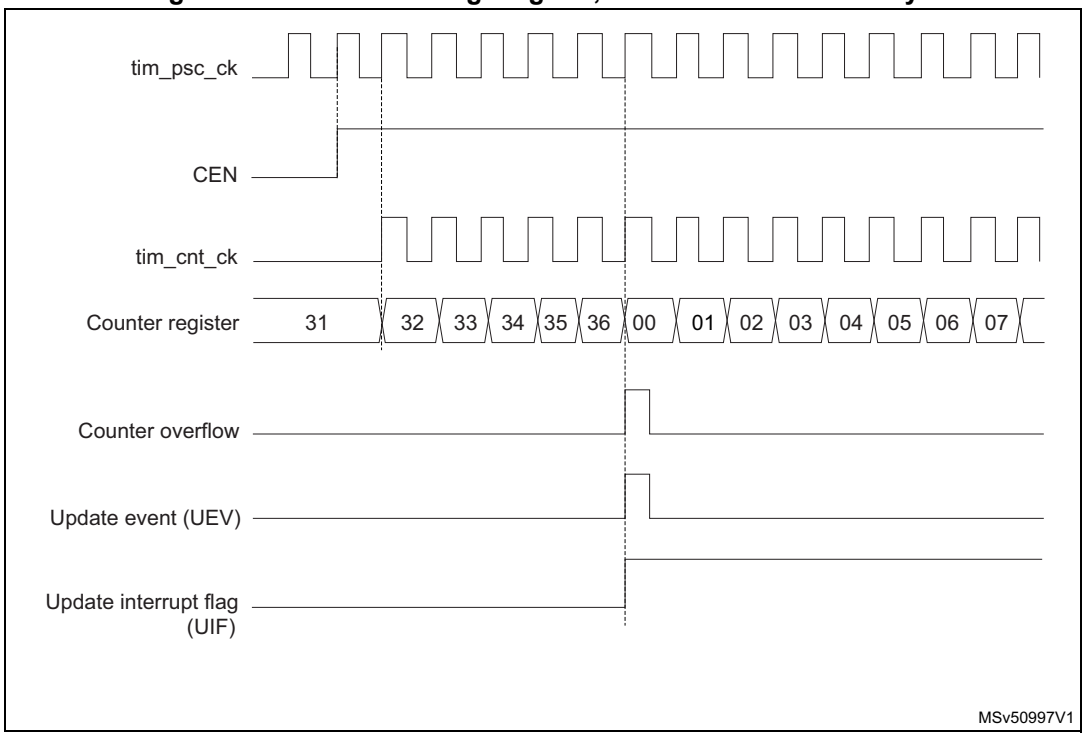


Figure 460. Counter timing diagram, internal clock divided by 2

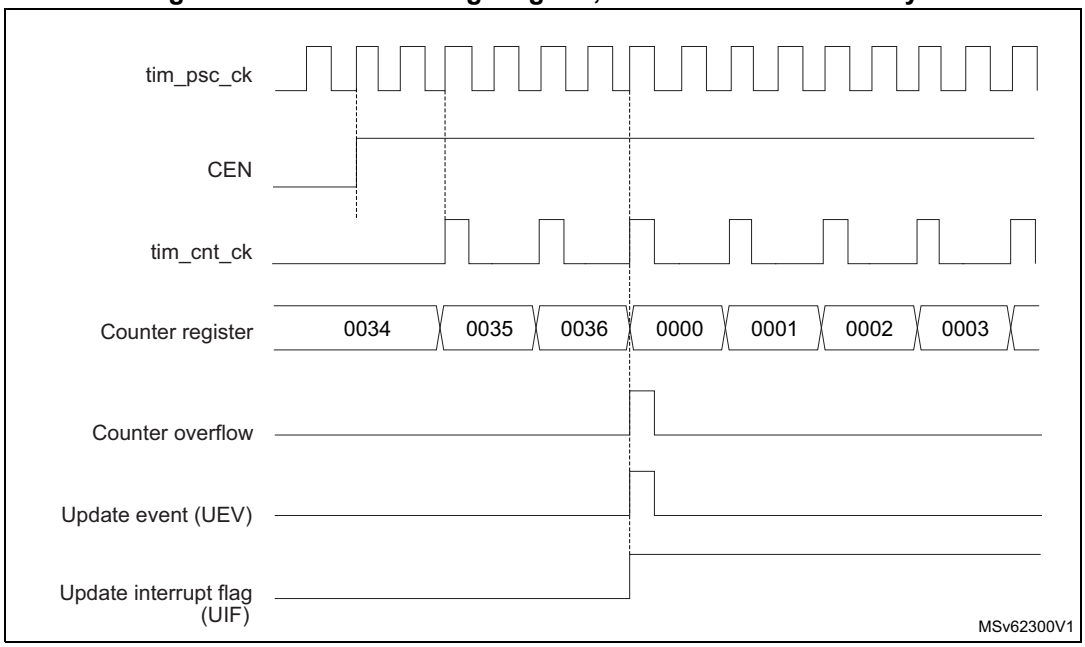


Figure 461. Counter timing diagram, internal clock divided by 4

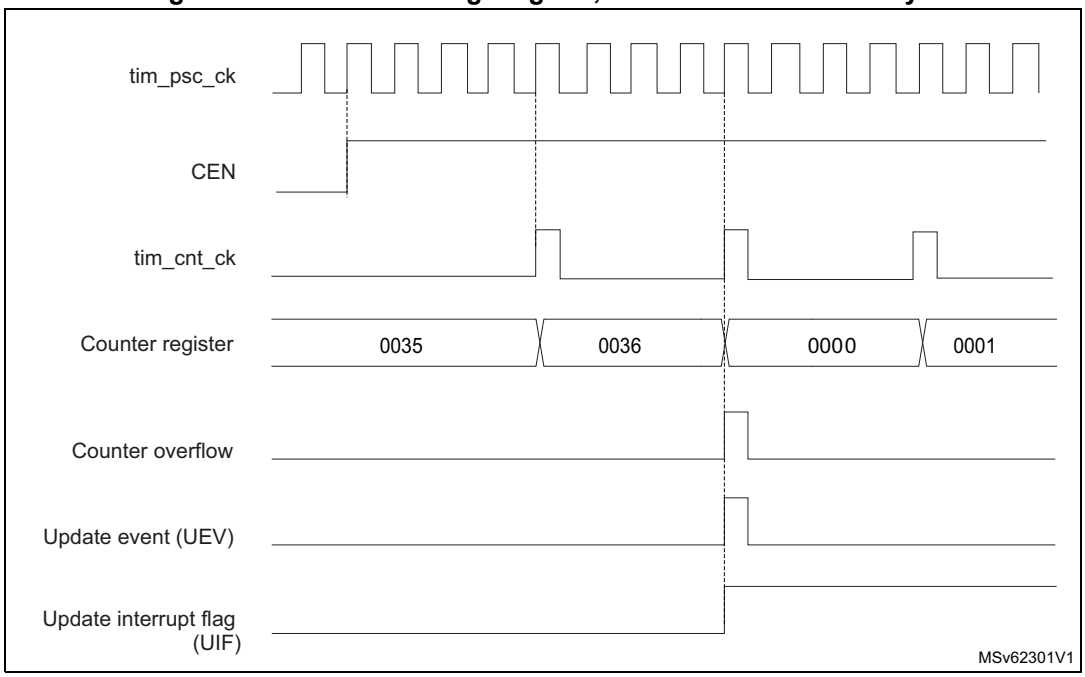


Figure 462. Counter timing diagram, internal clock divided by N

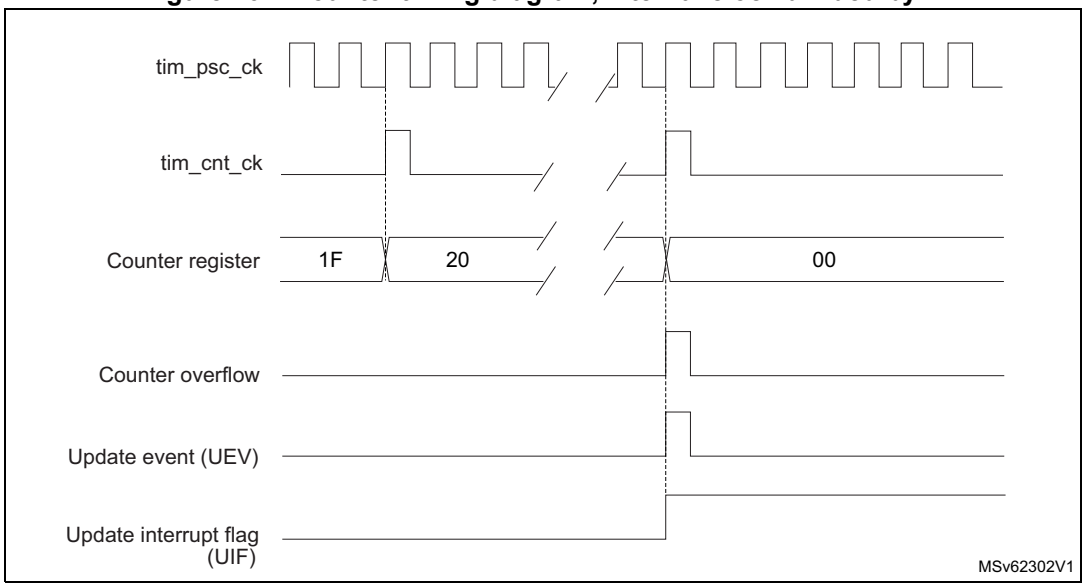


Figure 463. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

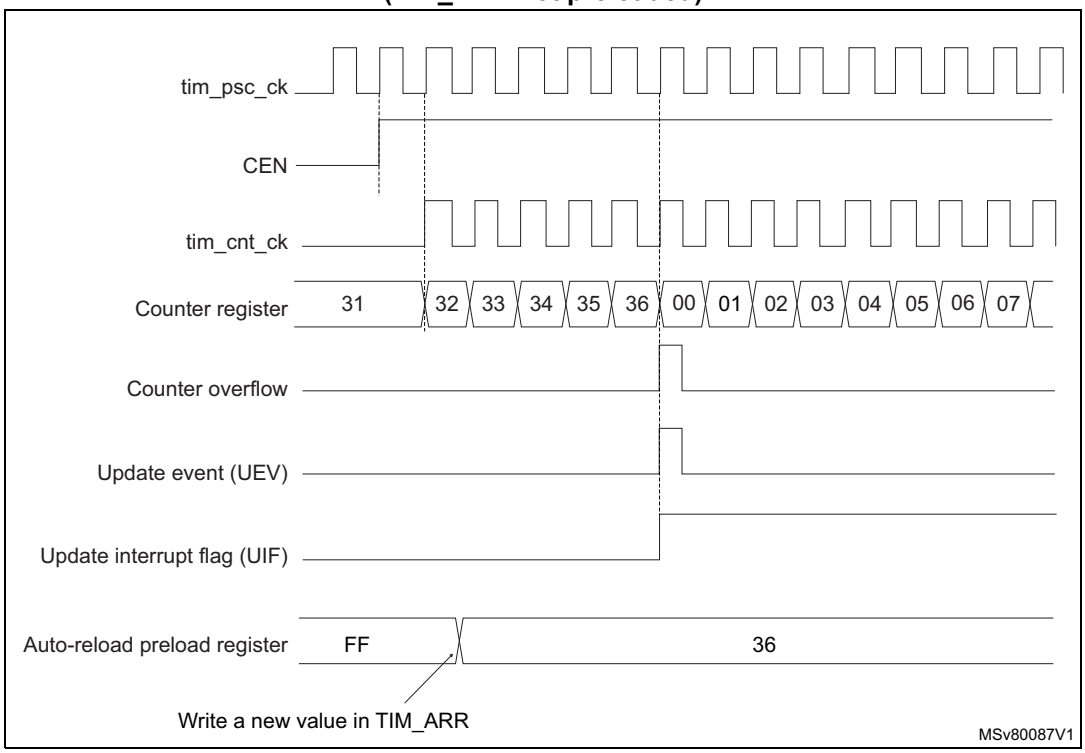
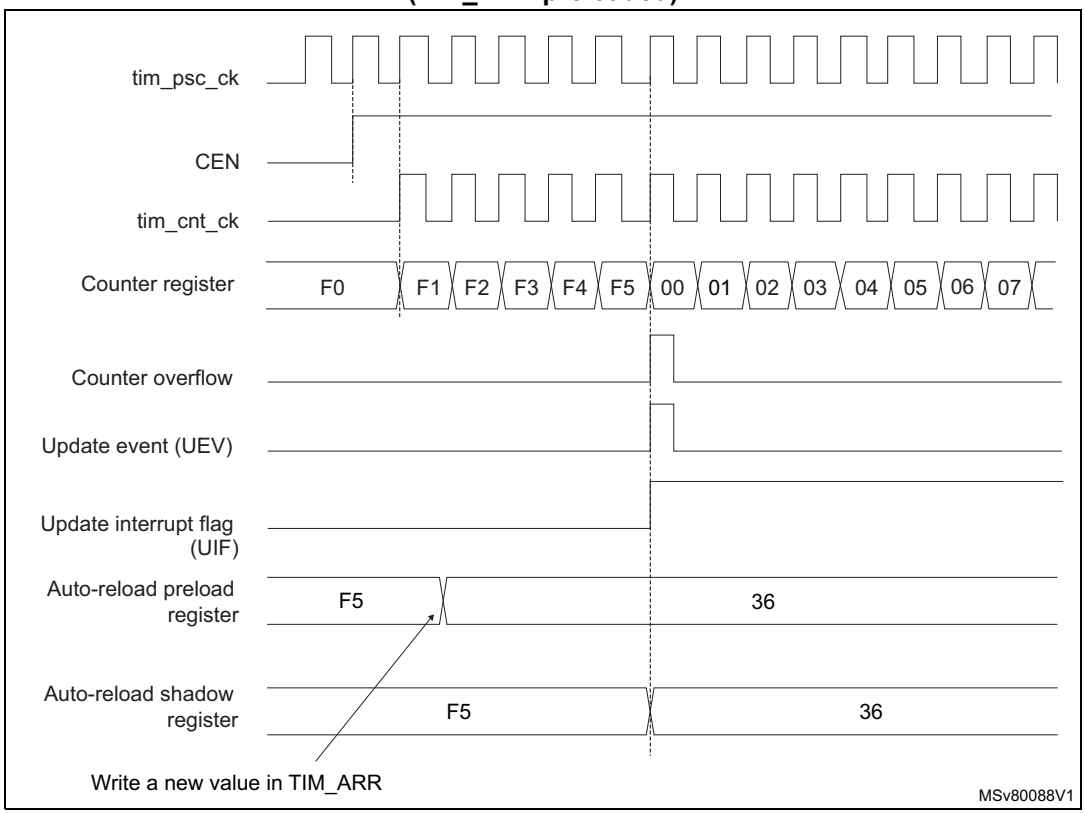


Figure 464. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 465. Counter timing diagram, internal clock divided by 1

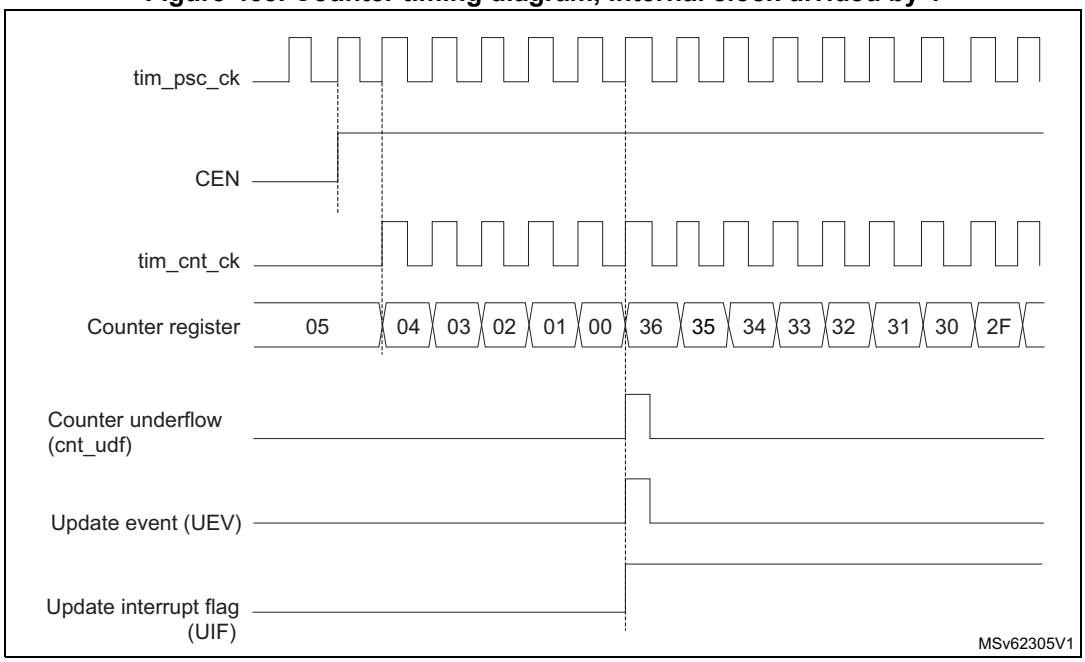


Figure 466. Counter timing diagram, internal clock divided by 2

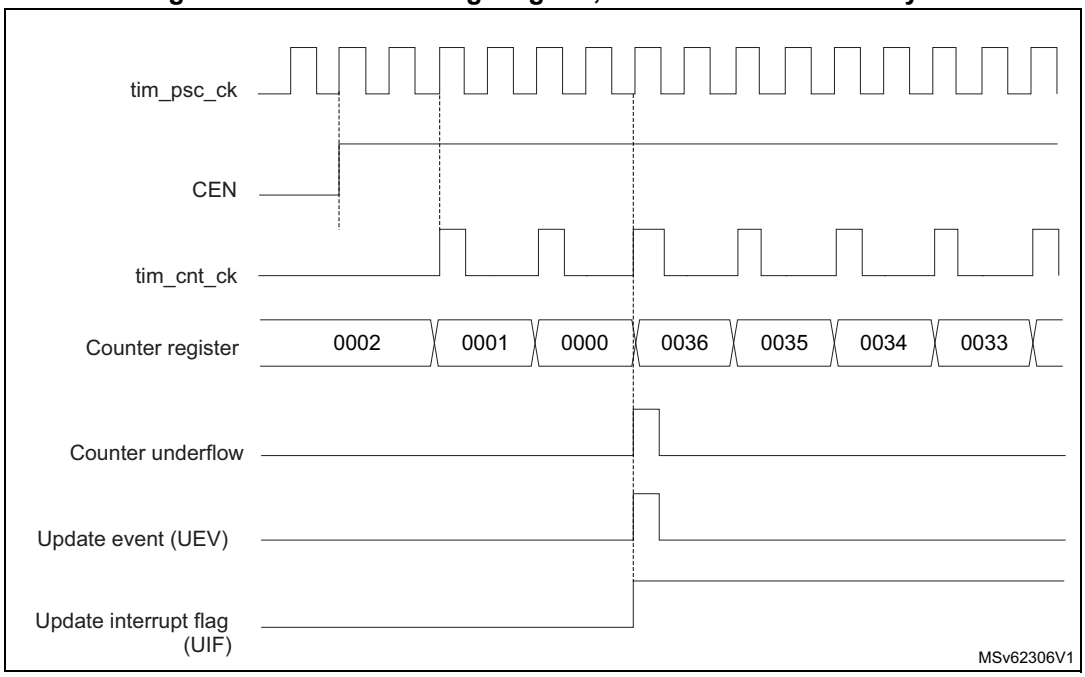


Figure 467. Counter timing diagram, internal clock divided by 4

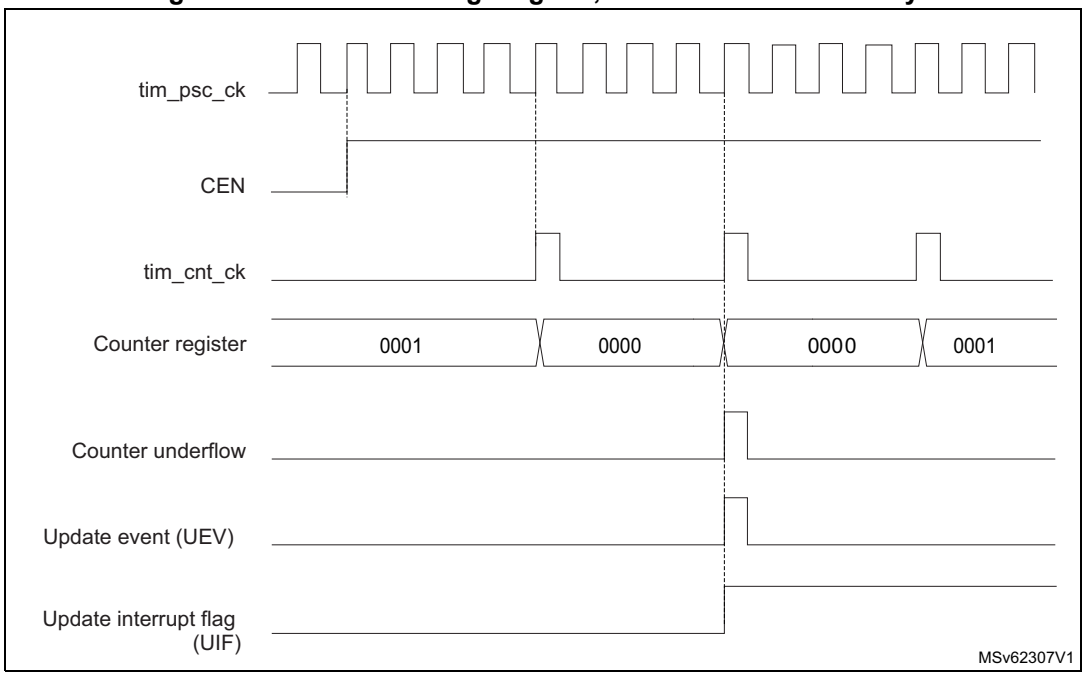


Figure 468. Counter timing diagram, internal clock divided by N

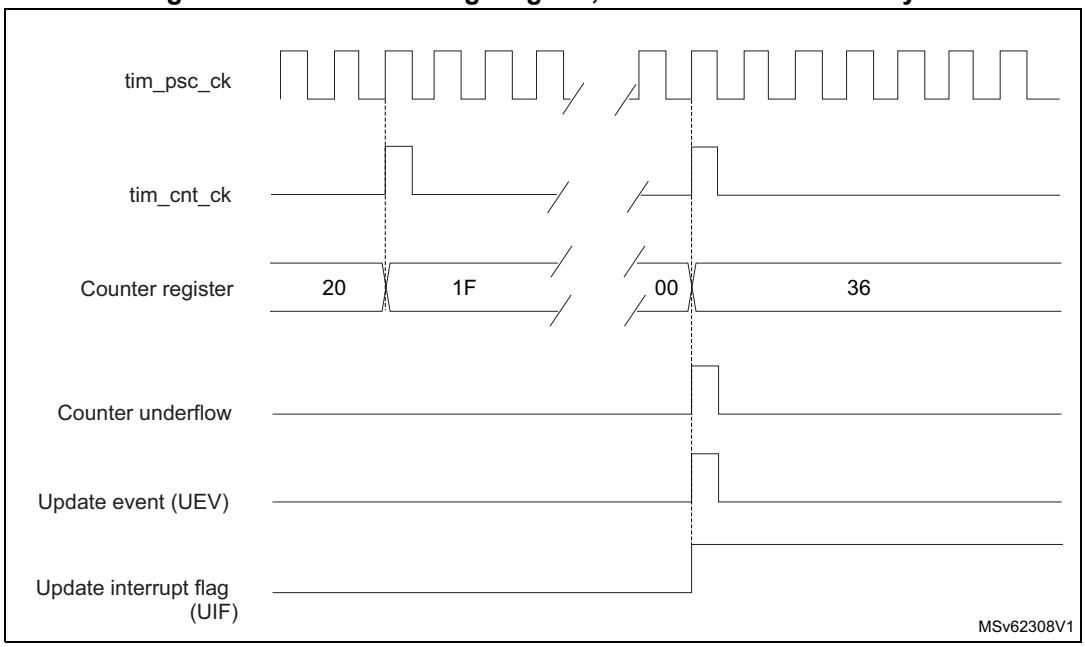
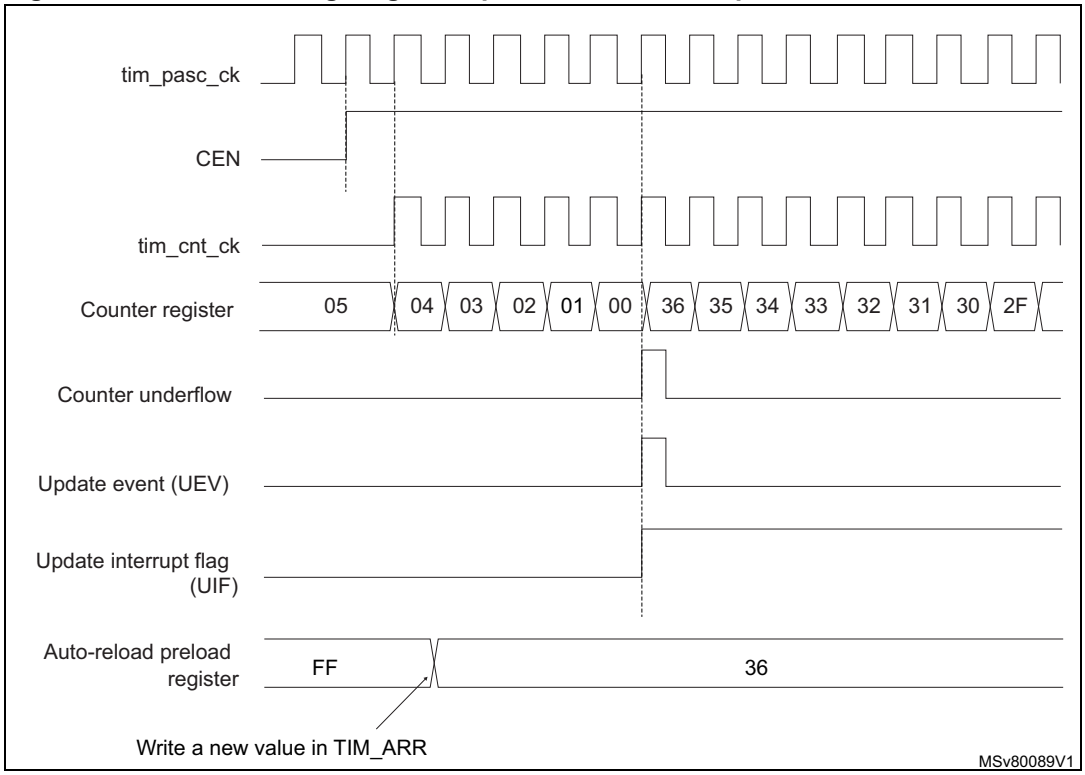


Figure 469. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIM_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

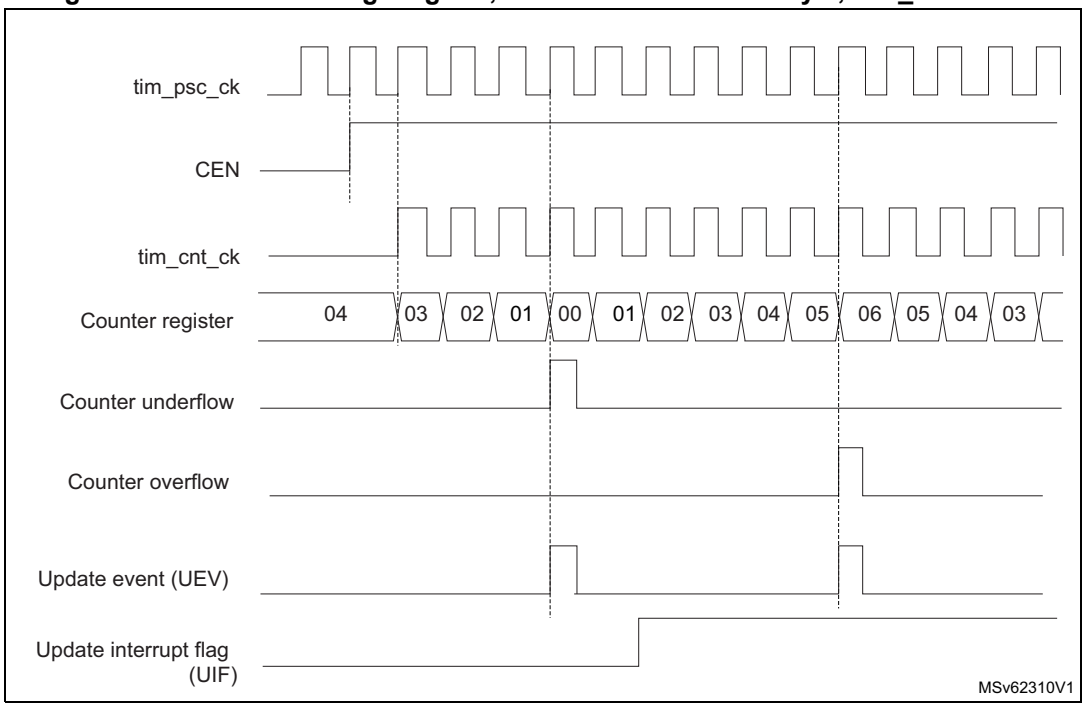
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register)
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 470. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 41.6: TIM1/TIM8/TIM20 registers*).

Figure 471. Counter timing diagram, internal clock divided by 2

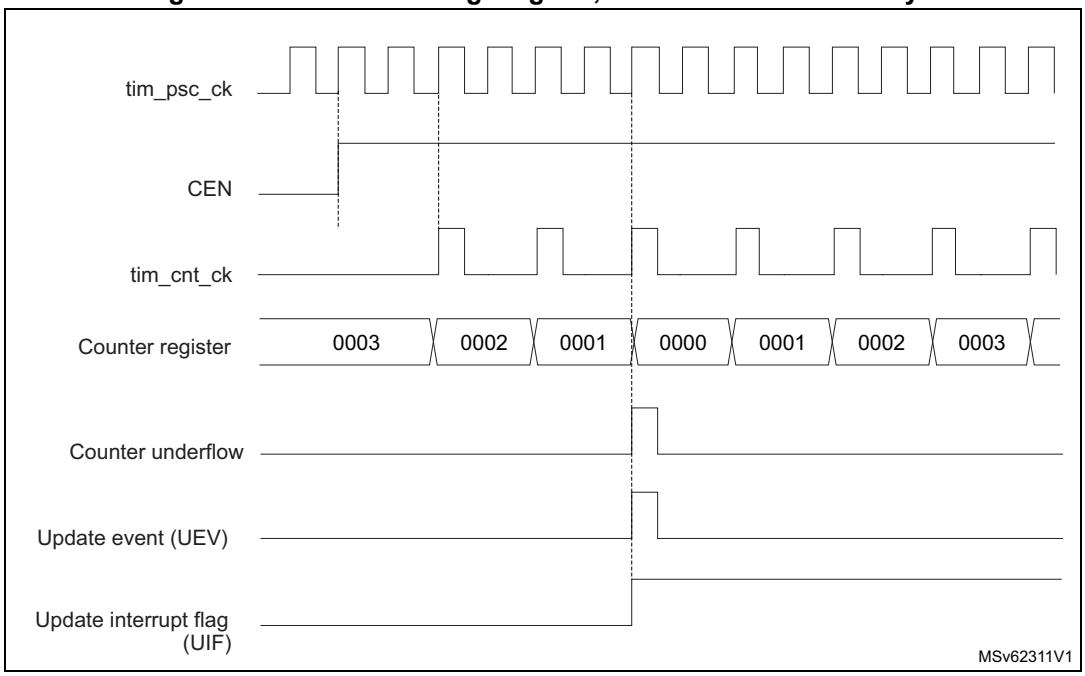


Figure 472. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36

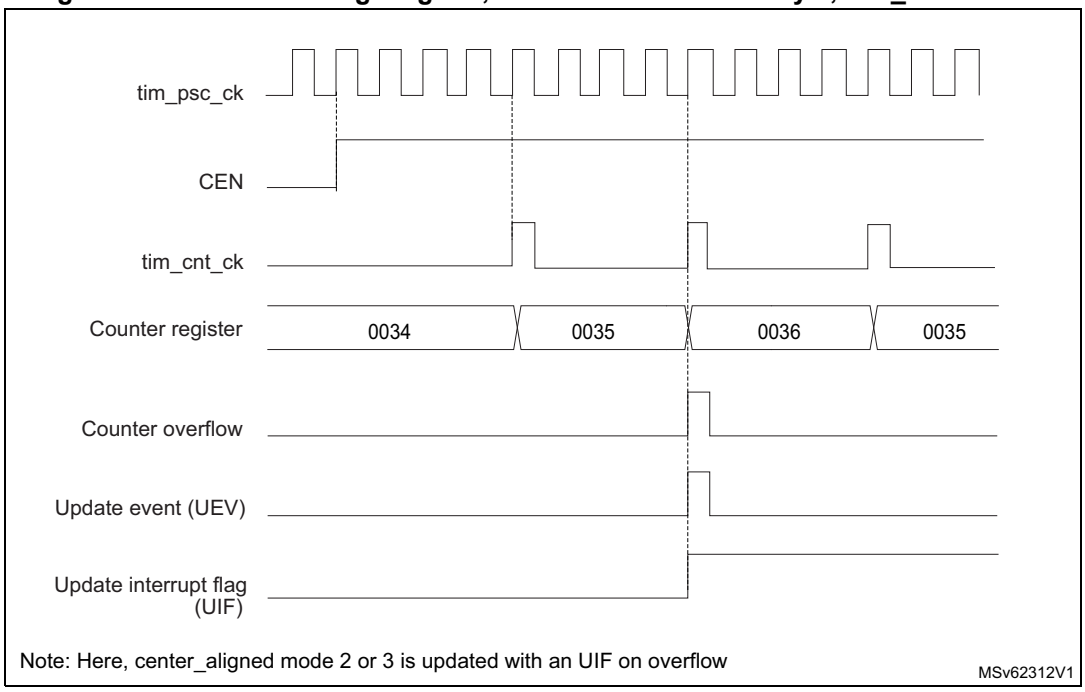


Figure 473. Counter timing diagram, internal clock divided by N

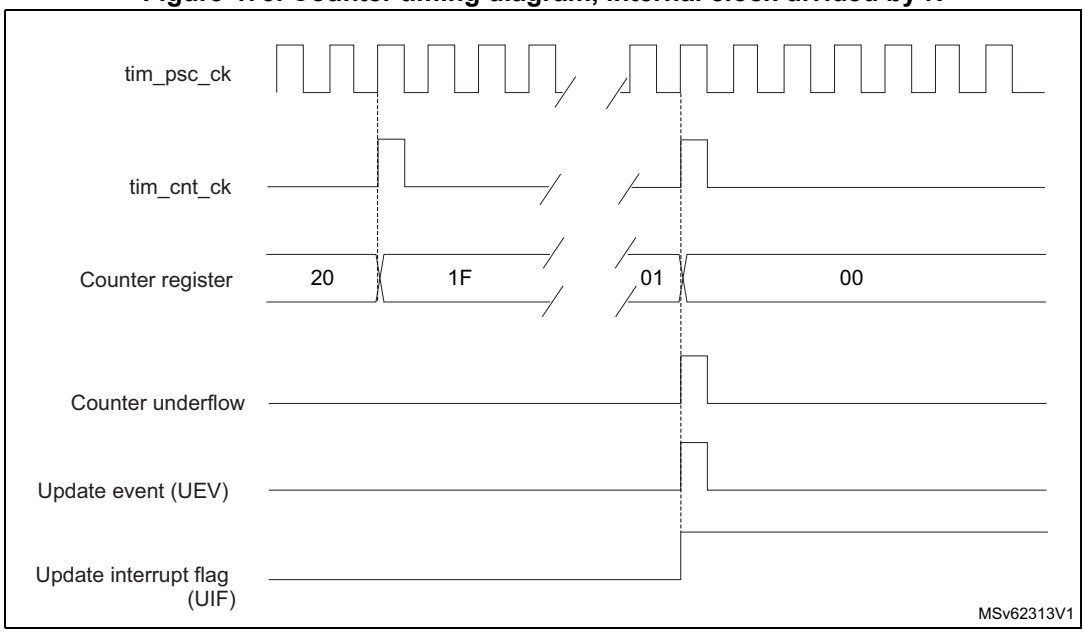


Figure 474. Counter timing diagram, update event with ARPE = 1 (counter underflow)

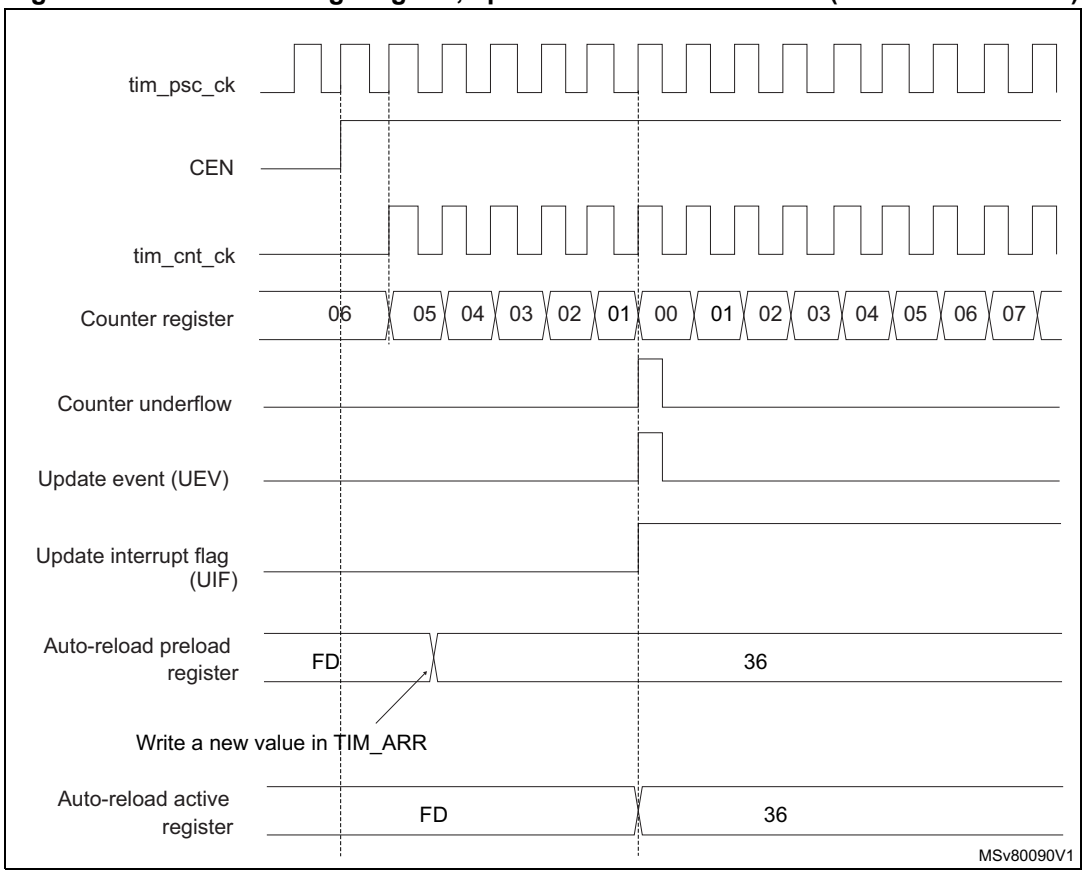
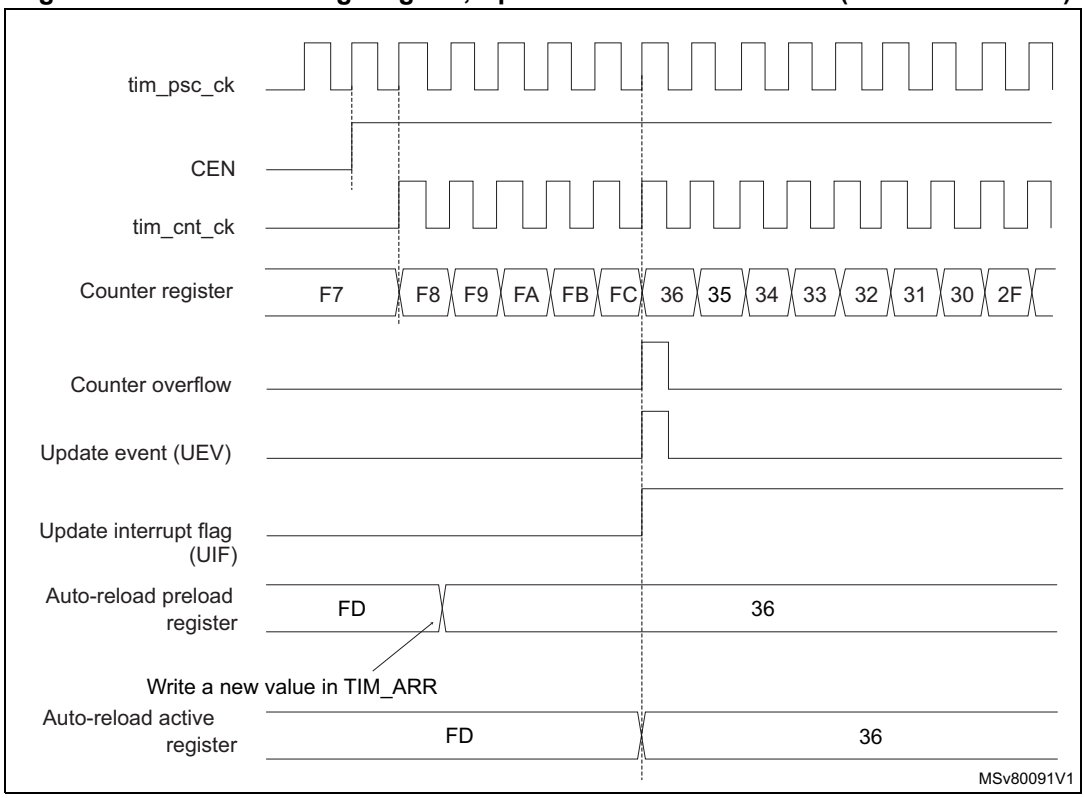


Figure 475. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



41.3.5 Repetition counter

Section 41.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIM_ARR autoreload register, TIM_PSC prescaler register, but also TIM_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIM_RCR repetition counter register.

The repetition counter is decremented:

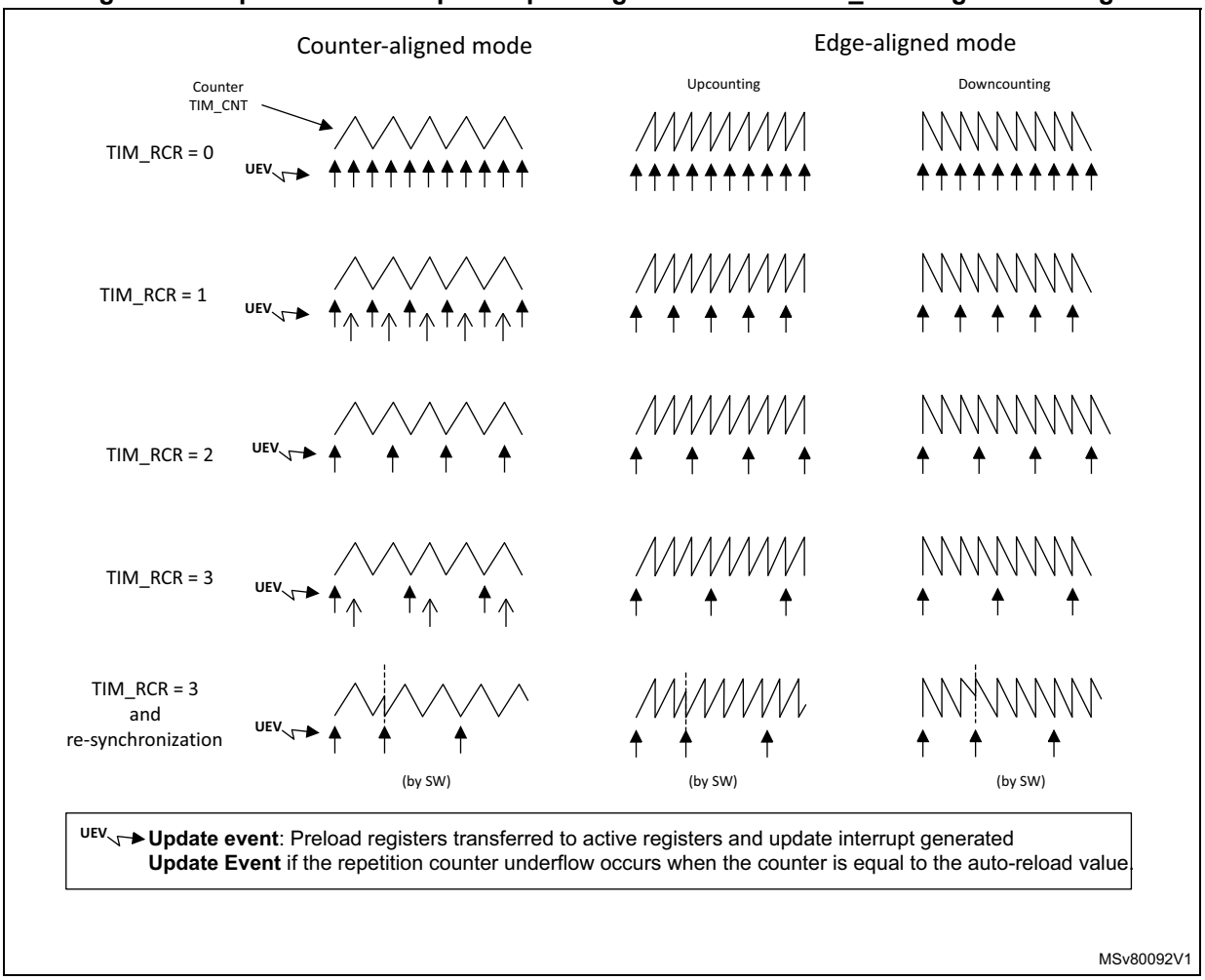
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIM_RCR register value (refer to *Figure 476*). When the update event is generated by software (by setting the UG bit in TIM_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIM_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 476. Update rate examples depending on mode and TIM_RCR register settings



41.3.6 External trigger input

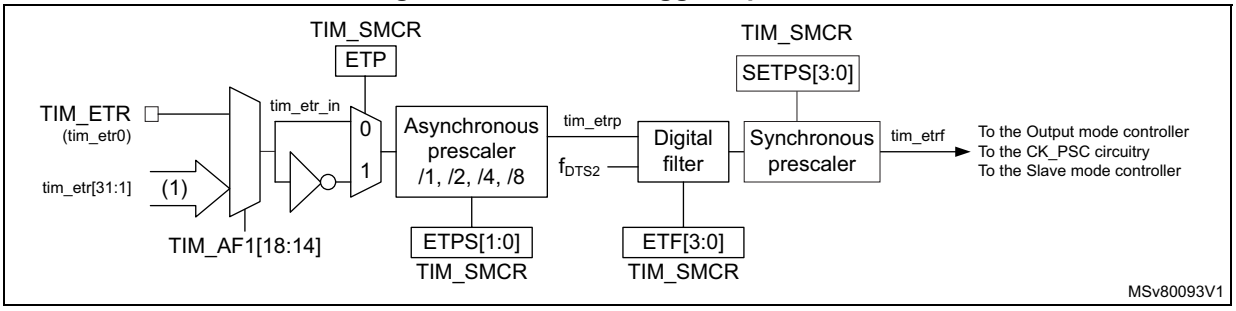
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 41.3.7*)
- trigger for the slave mode (see *Section 41.3.35*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 41.3.9*)

Figure 477 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIM_SMCR register. The trigger frequency can be decreased with the prescalers programmed by the ETPS[1:0] and SETPS[3:0] bitfields and digitally filtered as per the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an

external clock, to condition the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 477. External trigger input block



The external trigger input block features 2 prescalers with different functionalities:

- An asynchronous prescaler (division by 1, 2, 4 or 8), able to process tim_etr_in signals with frequencies above tim_ker_ck
- A linear synchronous prescaler (division by 1,2,3,..15), only able to operate with tim_etrp signal frequencies below tim_ker_ck/3

Note: When tim_etrf is used as PWM reset input for cycle-by-cycle current regulation (see Section 41.3.23), SETPS[3:0] must be set to 0 (no prescaling).

Note: The SETPS[3:0] bitfield is always preloaded: a new value will become active only once an update event has occured.

The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[4:0] bitfield in the TIM_AF1 register.

Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for the list of sources connected to the etr_in input in the product.

41.3.7 Clock selection

The counter clock can be provided by the following clock sources:

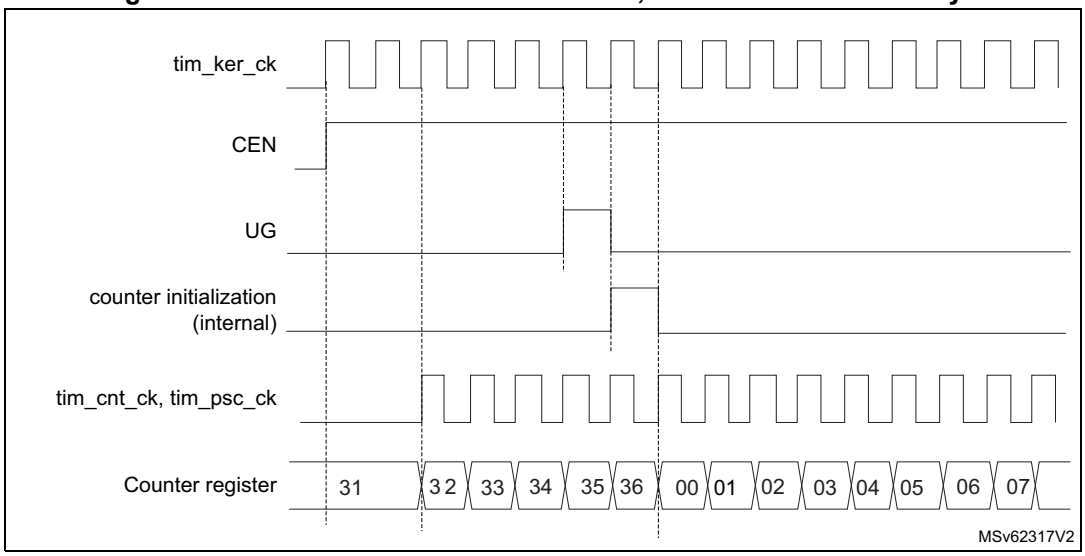
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 478 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

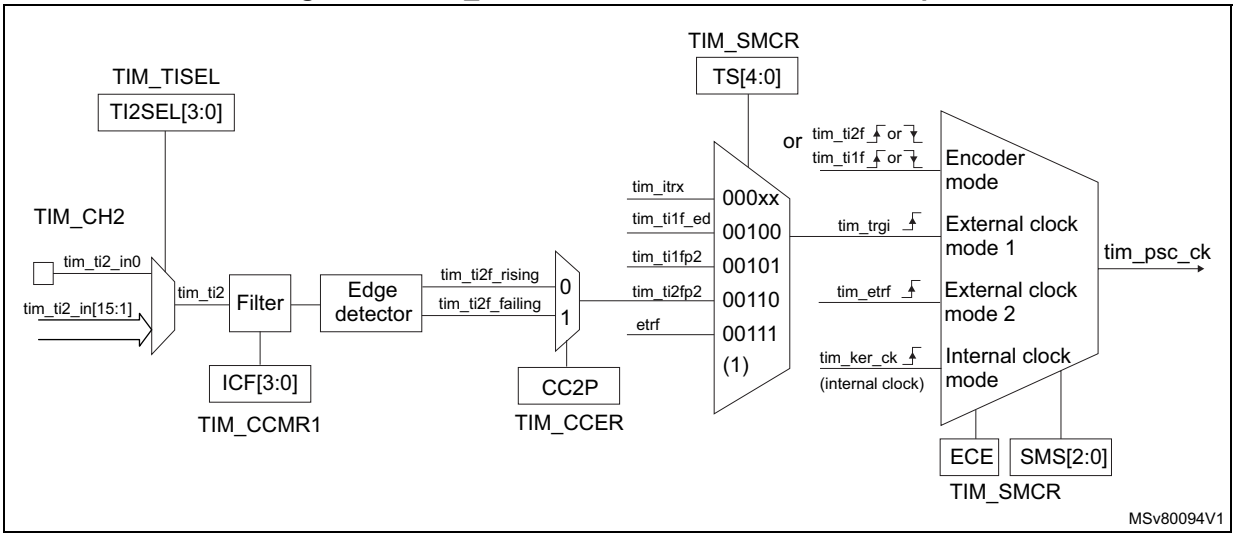
Figure 478. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 00111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 479. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

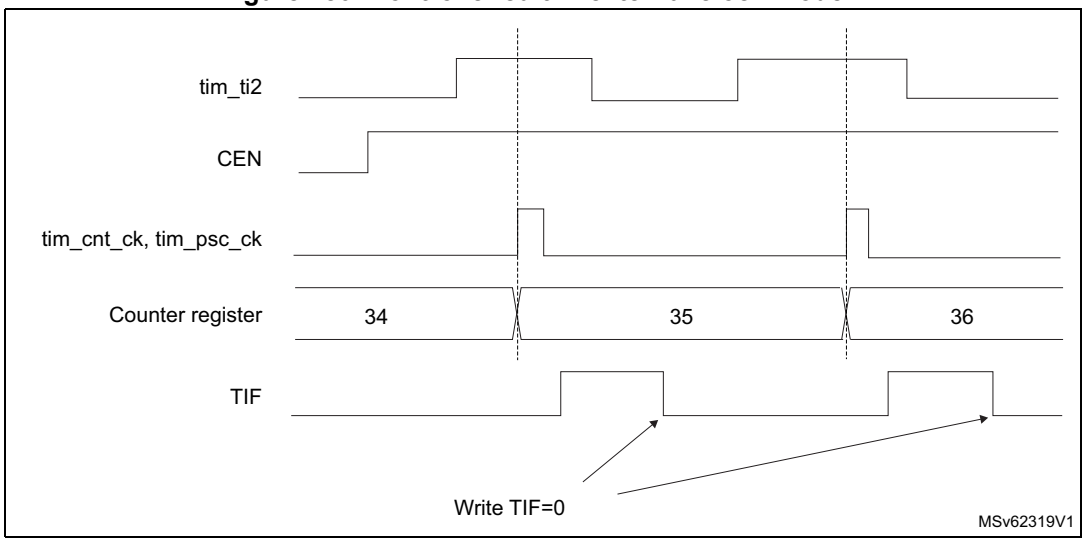
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIM_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 480. Control circuit in external clock mode 1



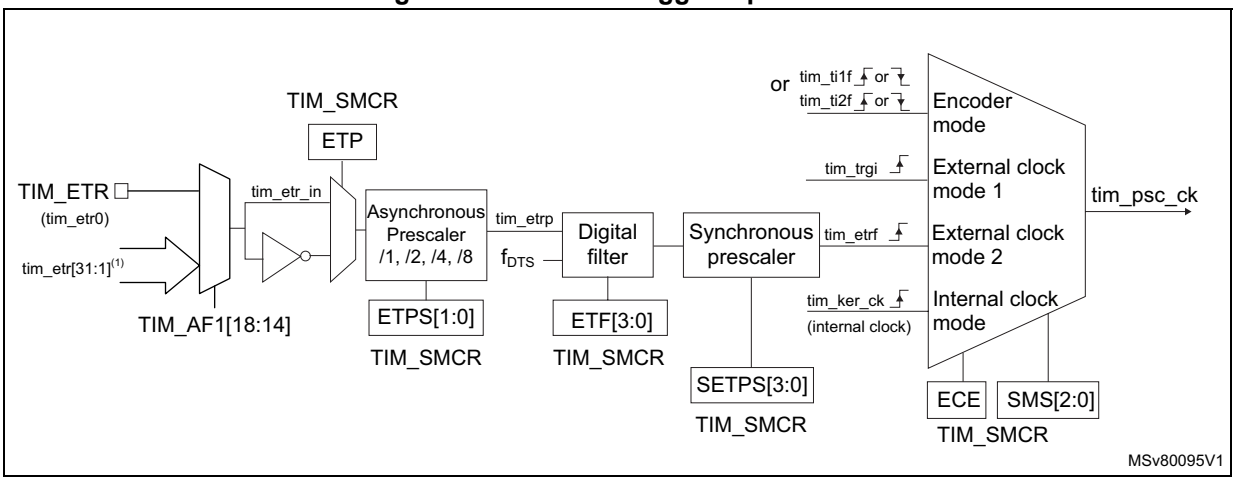
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIM_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 481* gives an overview of the external trigger input block.

Figure 481. External trigger input block



1. Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals*.

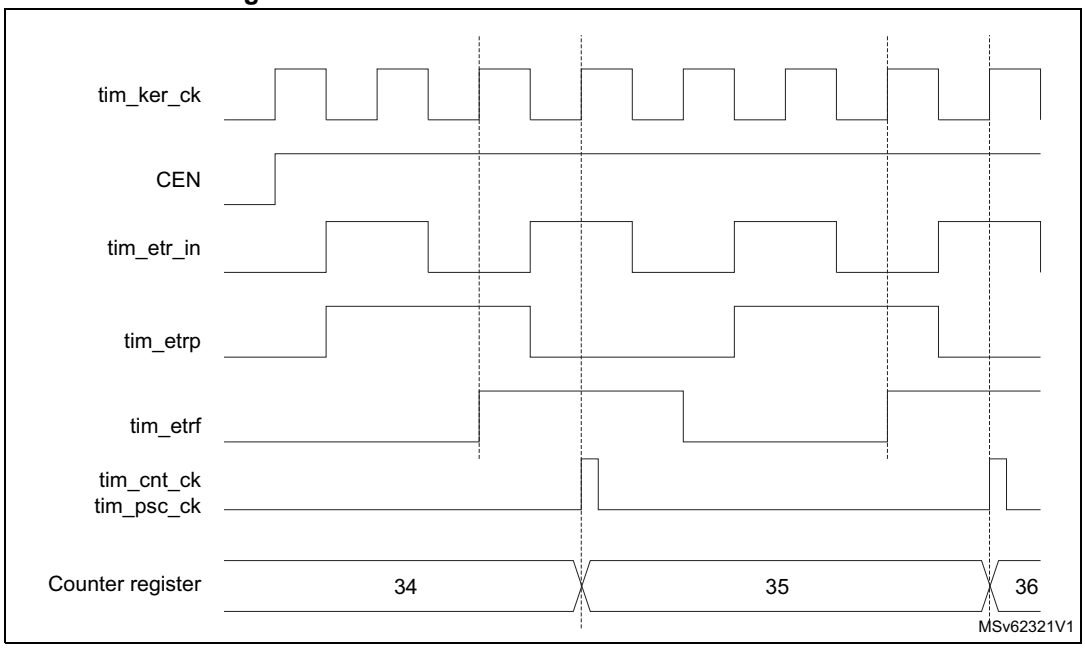
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIM_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 and SETPS[3:0] = 0 in the TIM_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP = 0 in the TIM_SMCR register
4. Enable external clock mode 2 by writing ECE = 1 in the TIM_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 482. Control circuit in external clock mode 2



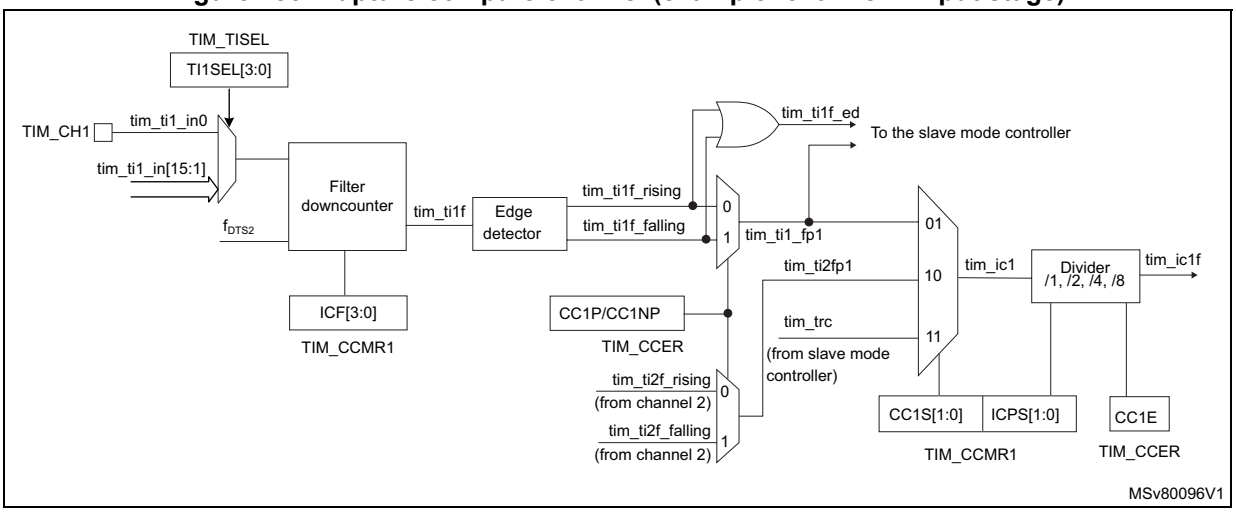
41.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 483 to *Figure 486* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 483. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 484. Capture/compare channel 1 main circuit

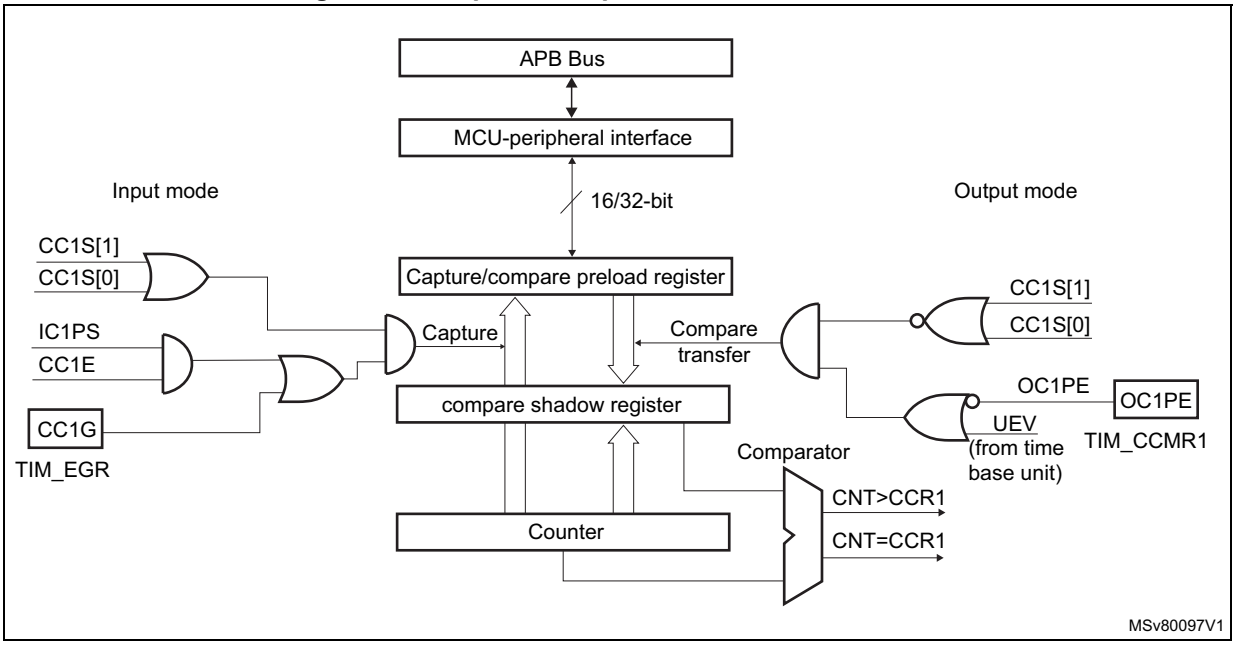
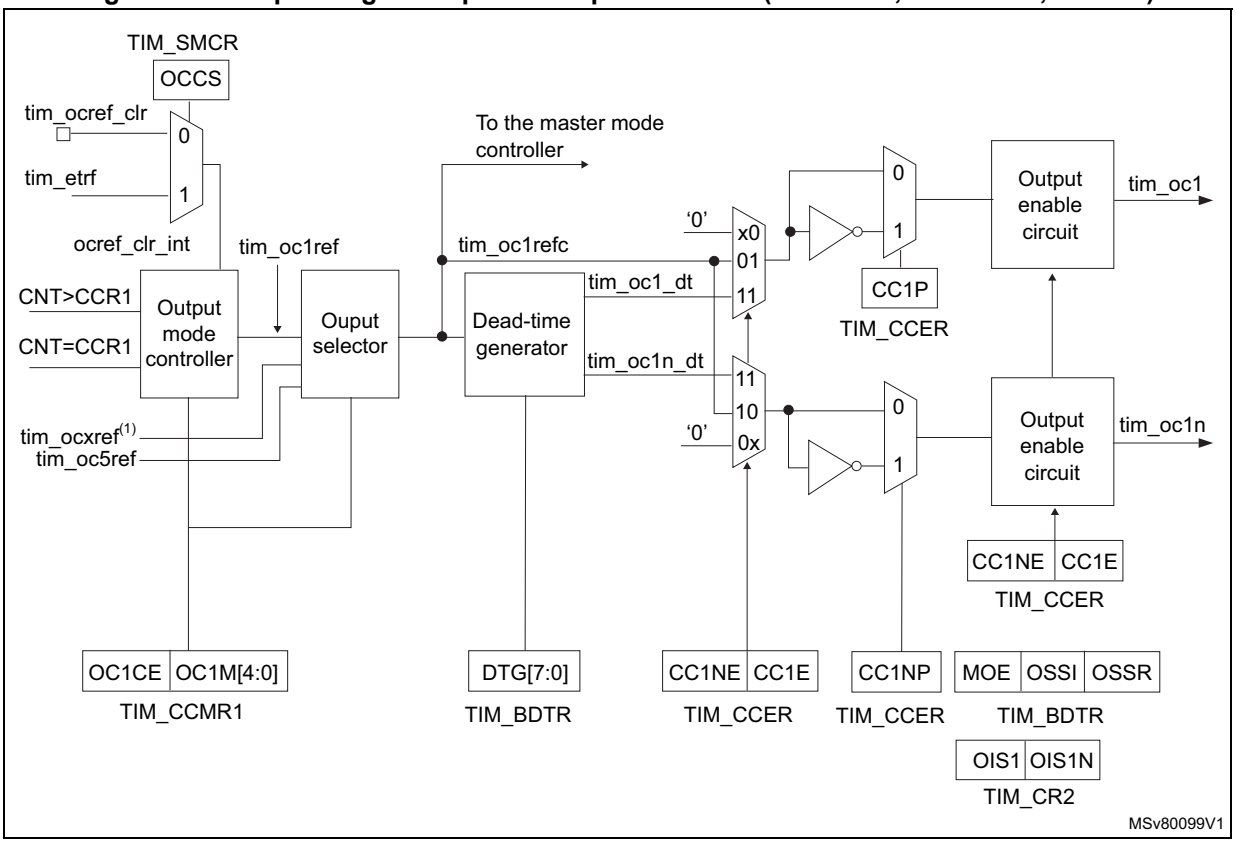
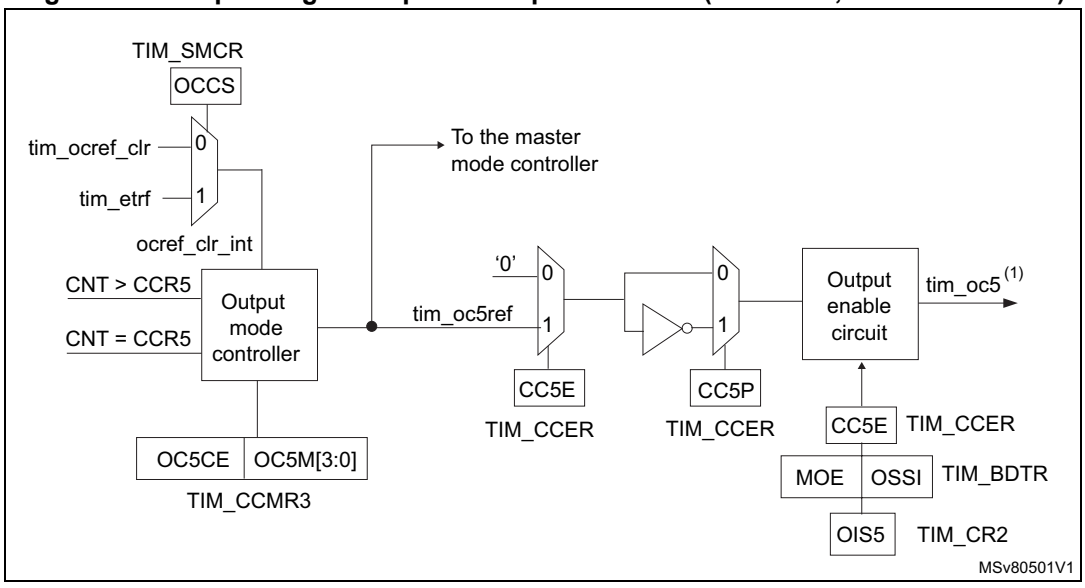


Figure 485. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 486. Output stage of capture/compare channel (channel 5, idem ch. 6 and 7)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

41.3.9 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIM_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS2} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.
- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIM_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

41.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges.

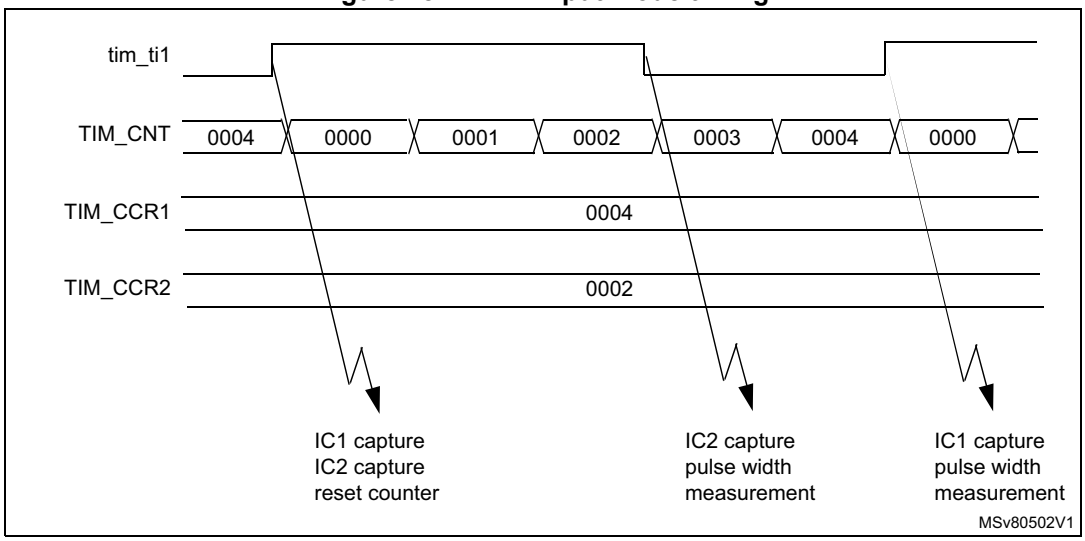
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIM_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 487. PWM input mode timing



41.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

41.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 to 7 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 00000), be set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

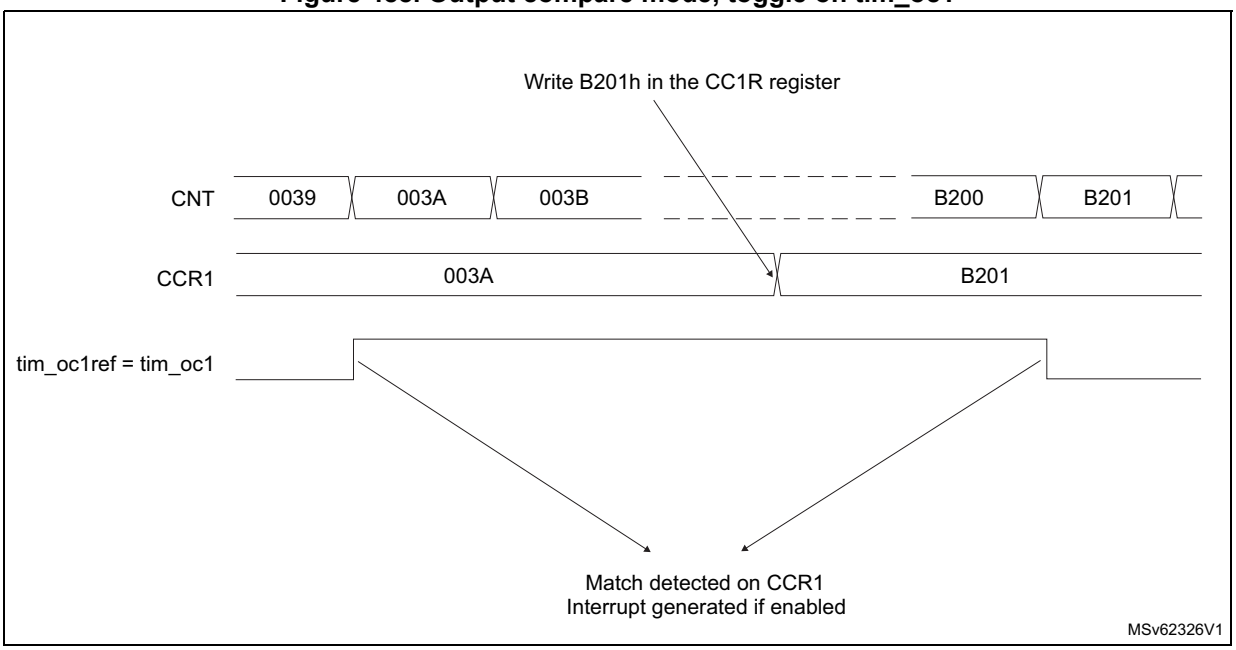
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 488*.

Figure 488. Output compare mode, toggle on tim_oc1



41.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIM_CCER and TIM_BDTR registers). Refer to the TIM_CCER register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

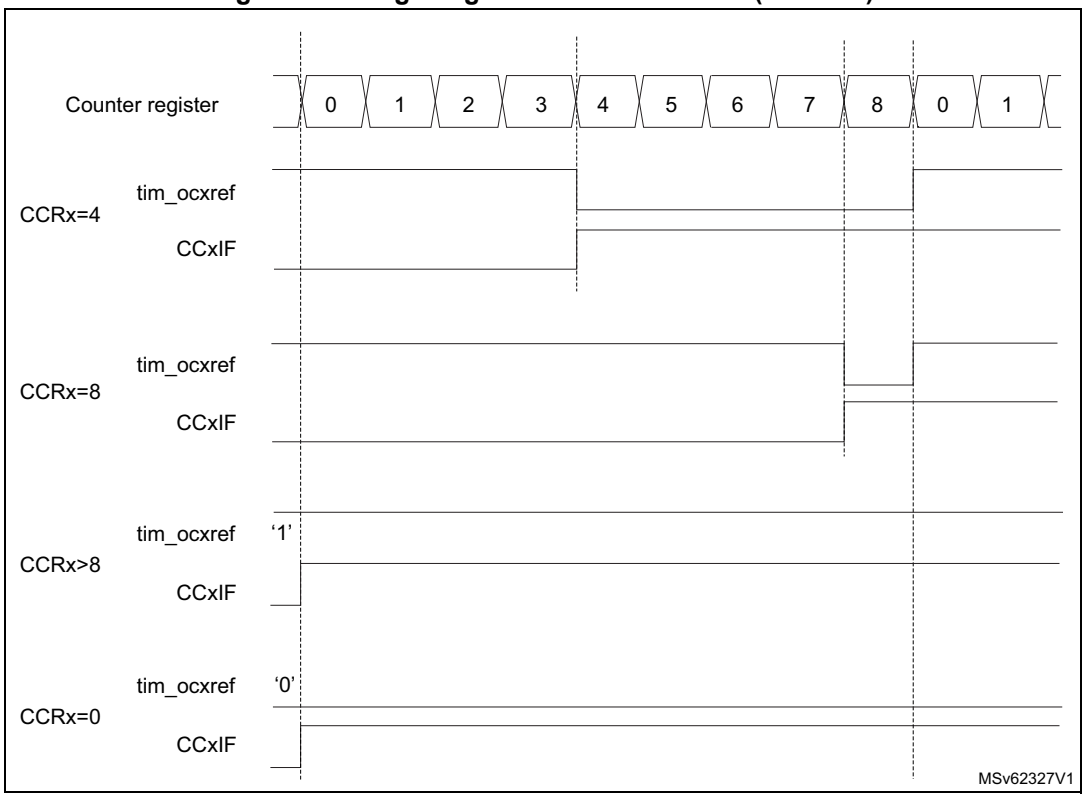
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 489* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 489. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIM_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT > TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

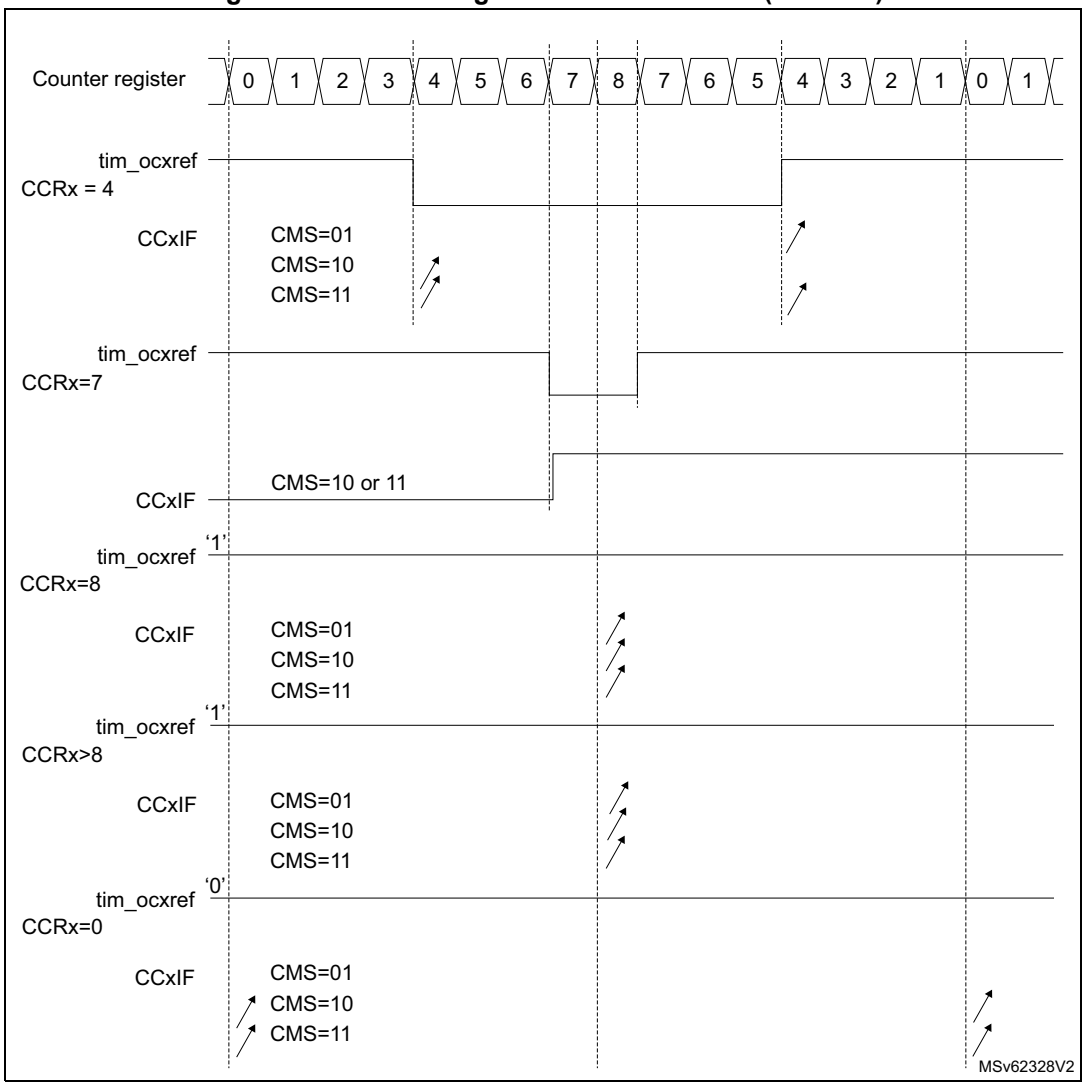
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 490 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 490. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

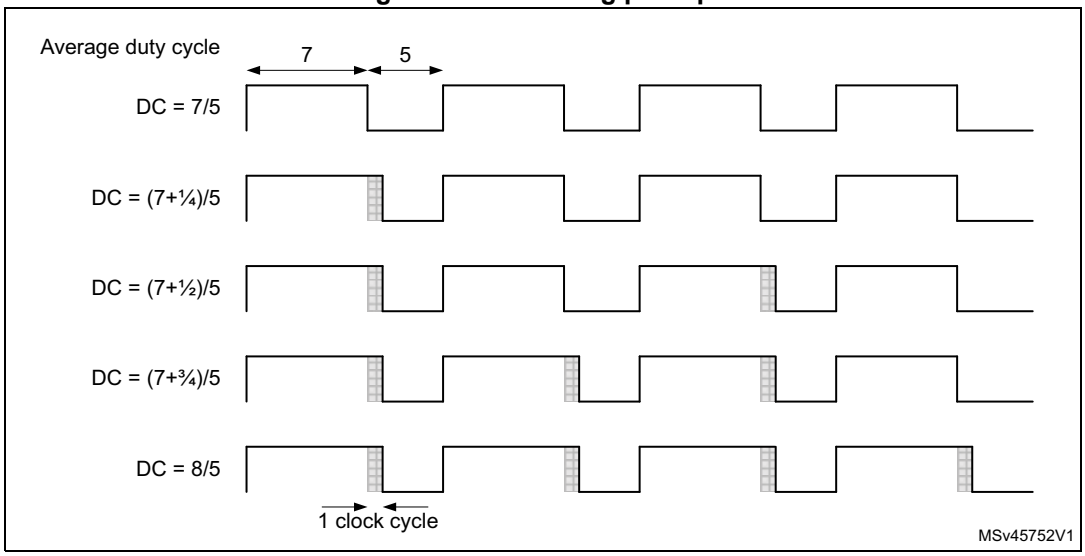
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT > TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 491* presents the dithering principle applied to four consecutive PWM cycles.

Figure 491. Dithering principle



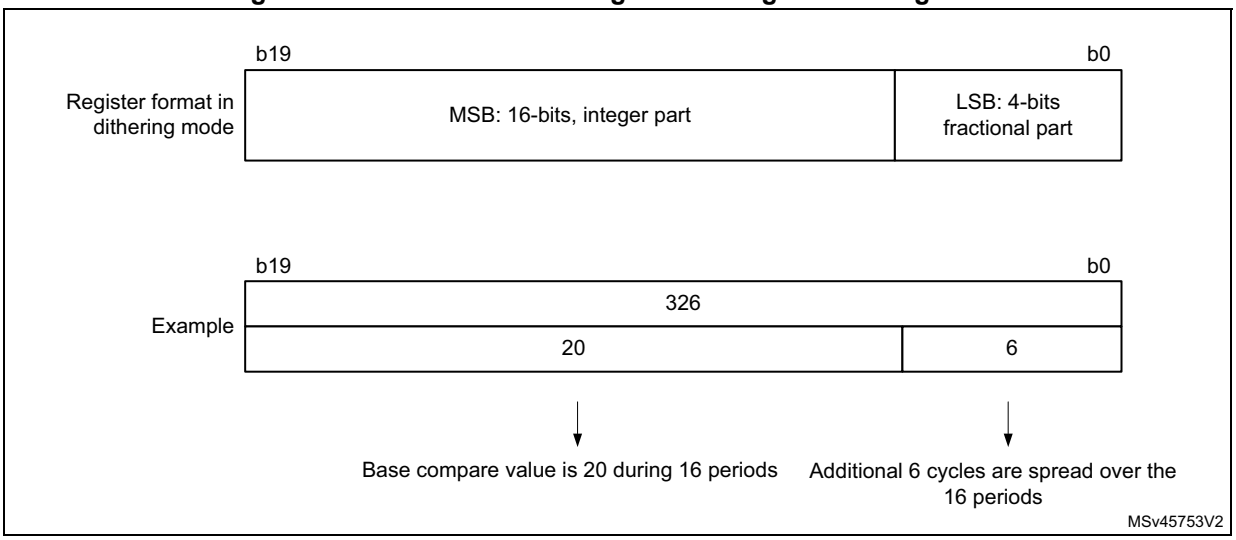
When the dithering mode is enabled, the register coding is changed as follows (see *Figure 492* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 492. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

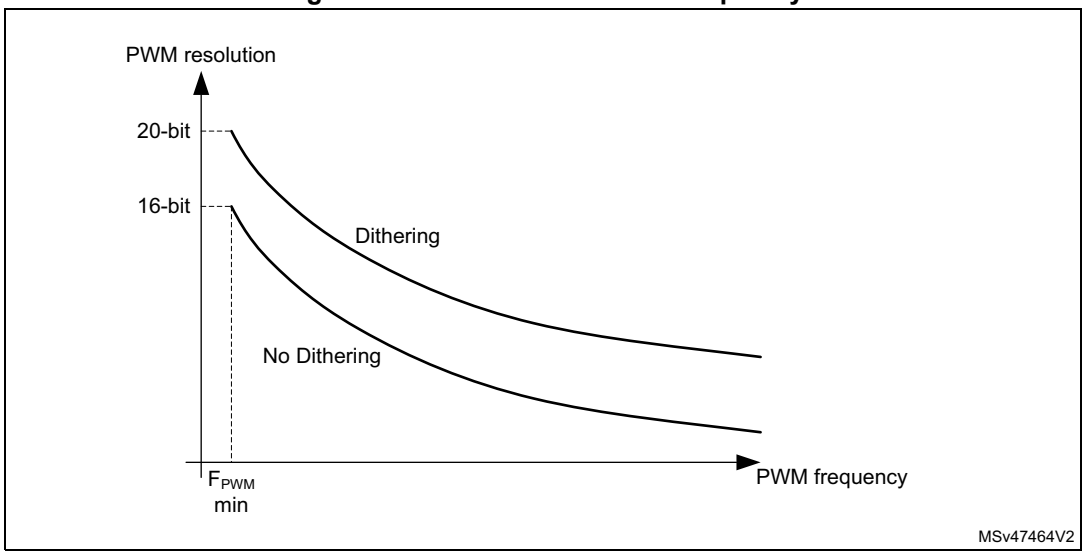
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

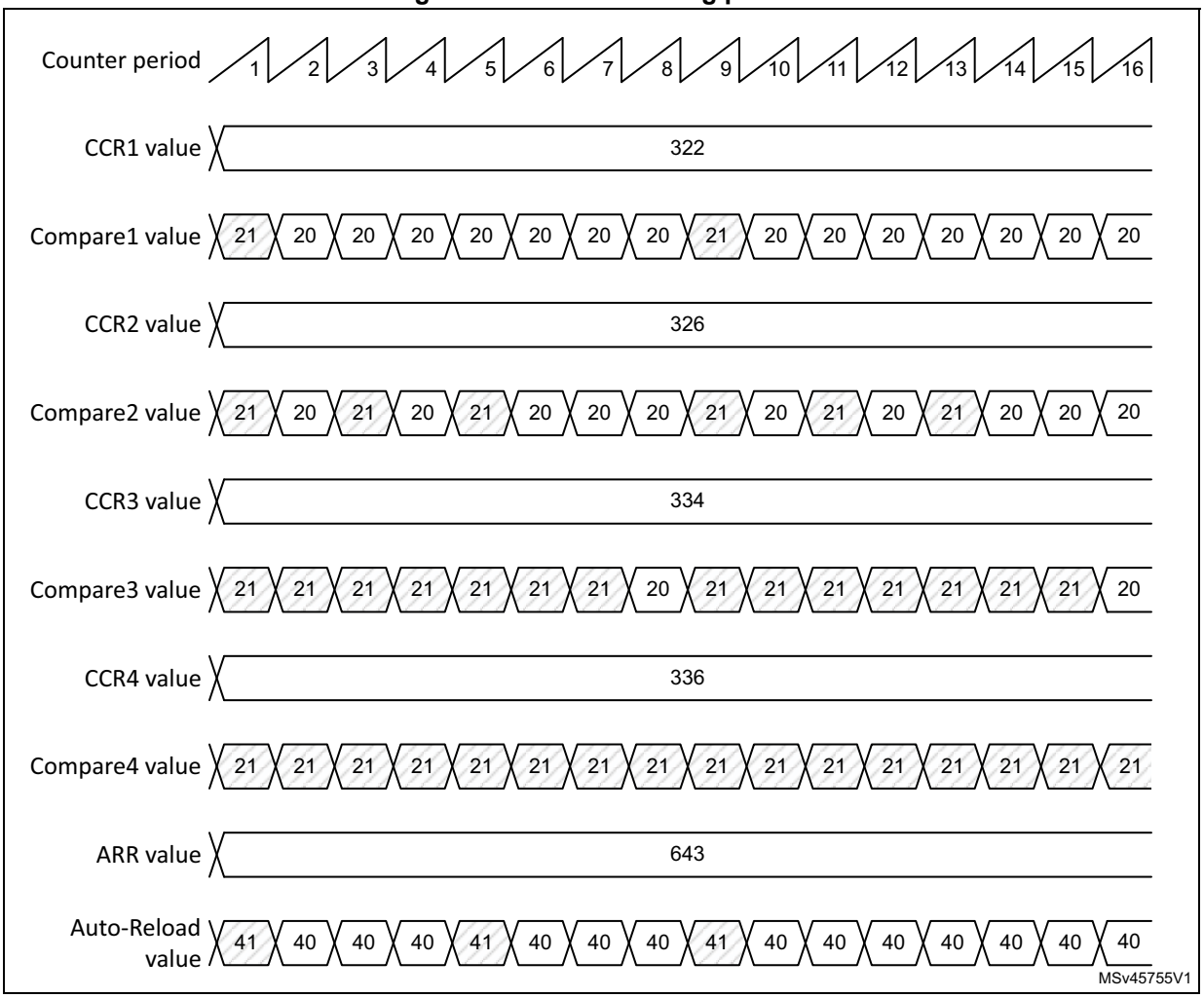
As shown on *Figure 493*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 493. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 494*.

Figure 494. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 462*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 462. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 462. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 495*.

Figure 495. Dithering effect on duty cycle in center-aligned PWM mode

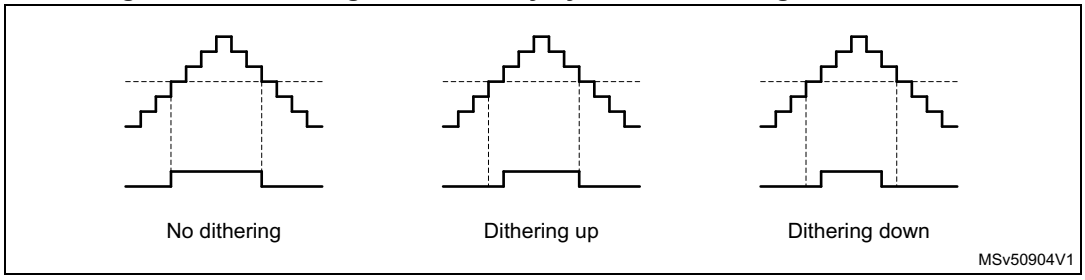


Table 463 shows how the dithering pattern is added in center-aligned PWM mode.

Table 463. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 463. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

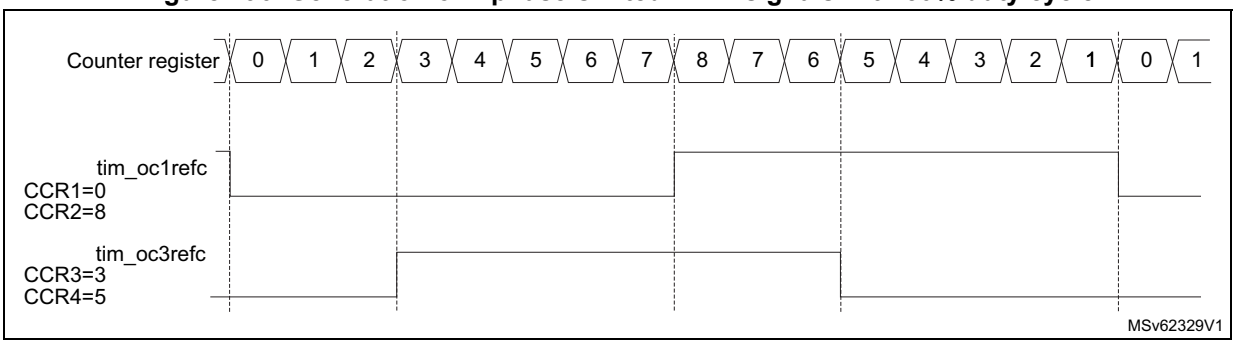
41.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle, for instance:

- tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2
- tim_oc3refc is controlled by TIM_CCR3 and TIM_CCR4
- ...

Figure 496 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 496. Generation of 2 phase-shifted PWM signals with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 464*. The selection is done

with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (e.g. Asymmetric PWM mode 3 and 4), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 to 6, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 00110 for the asymmetric PWM mode 1 on oc1refc.

For the asymmetric PWM modes 7 to 10, it is mandatory to have opposite PWM mode for both channels (one being in PWM mode 1 and the other in PWM mode 2). Refer to the *Table 464* for configuration.

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2 or a tim_oc2refc signal resulting from asymmetric PWM mode 1.

Table 464. Asymmetric PWM modes

<table>
<tr><th>Output compare mode</th><th>OCxM[4:0]</th><th>oc1refc</th><th>oc2refc</th><th>oc3refc</th><th>oc4refc</th><th>oc5refc</th><th>oc6refc</th><th>oc7refc</th><th>PWM Mode (1)</th></tr>
<tr><td>Asymmetric PWM mode 1</td><td>01110</td><td colspan="2" rowspan="2">oc1ref (up-counting)
oc2ref (down-counting)</td><td colspan="2" rowspan="2">oc3ref (up-counting)
oc4ref (down-counting)</td><td colspan="2" rowspan="2">oc5ref (up-counting)
oc6ref (down-counting)</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 2</td><td>01111</td><td>2</td></tr>
<tr><td>Asymmetric PWM mode 3</td><td>10010</td><td>oc1ref (up-counting)
oc5ref (down-counting)</td><td>oc2ref (up-counting)
oc5ref (down-counting)</td><td>oc3ref (up-counting)
oc5ref (down-counting)</td><td>oc4ref (up-counting)
oc5ref (down-counting)</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 4</td><td>10011</td><td>2</td></tr>
<tr><td>Asymmetric PWM mode 5</td><td>10100</td><td>oc1ref (up-counting)
oc6ref (down-counting)</td><td>oc2ref (up-counting)
oc6ref (down-counting)</td><td>oc3ref (up-counting)
oc6ref (down-counting)</td><td>oc4ref (up-counting)
oc6ref (down-counting)</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td>1</td></tr>
<tr><td>Asymmetric PWM mode 6</td><td>10101</td><td>2</td></tr>
</table>

Table 464. Asymmetric PWM modes

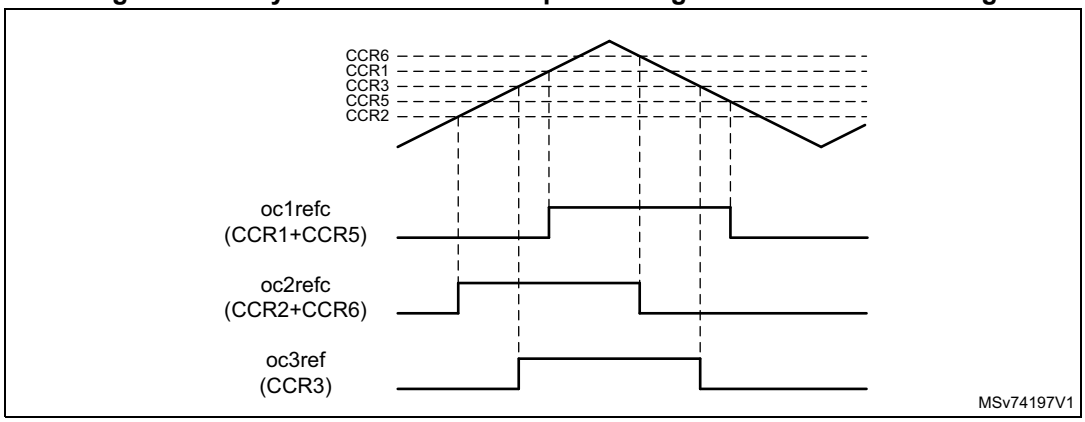
Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	oc5refc	oc6refc	oc7refc	PWM Mode (1)
Asymmetric PWM mode 7	10110	oc1ref OR oc5ref (up-counting) oc1refc=1 (down-counting)	oc2ref OR oc6ref (up-counting) oc2refc=1 (down-counting)	oc3ref OR oc4ref (up-counting) oc3refc=1 (down-counting)	oc4ref OR oc7ref (up-counting) oc4refc=1 (down-counting)	-	-	-	1(2)
Asymmetric PWM mode 8	10111	oc1ref AND oc5ref (up-counting) oc1refc=0 (down-counting)	oc2ref AND oc6ref (up-counting) oc2refc=0 (down-counting)	oc3ref AND oc4ref (up-counting) oc3refc=0 (down-counting)	oc4ref AND oc7ref (up-counting) oc4refc=0 (down-counting)				2(3)
Asymmetric PWM mode 9	11000	oc1refc=1 (up-counting) oc1ref OR oc5ref (down-counting)	oc2refc=1 (up-counting) oc2ref OR oc6ref (down-counting)	oc3refc=1 (up-counting) oc3ref OR oc4ref (down-counting)	oc4refc=1 (up-counting) oc4ref OR oc7ref (down-counting)	-	-	-	1(2)
Asymmetric PWM mode 10	11001	oc1refc=0 (up-counting) oc1ref AND oc5ref (down-counting)	oc2refc=0 (up-counting) oc2ref AND oc6ref (down-counting)	oc3refc=0 (up-counting) oc3ref AND oc4ref (down-counting)	oc4refc=0 (up-counting) oc7ref AND oc4ref (down-counting)				2(3)

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the 2 channels used for the asymmetric mode
2. oc5ref, oc6ref, oc4ref and oc7ref must be programmed in PWM mode 2
3. oc5ref, oc6ref, oc4ref and oc7ref must be programmed in PWM mode 1

The *Figure 497* represents an example suitable for single-shunt current sensing techniques requiring phase-shifted PWMs, for 3-phase motor control. This is done using asymmetric PWM mode 3 and 5:

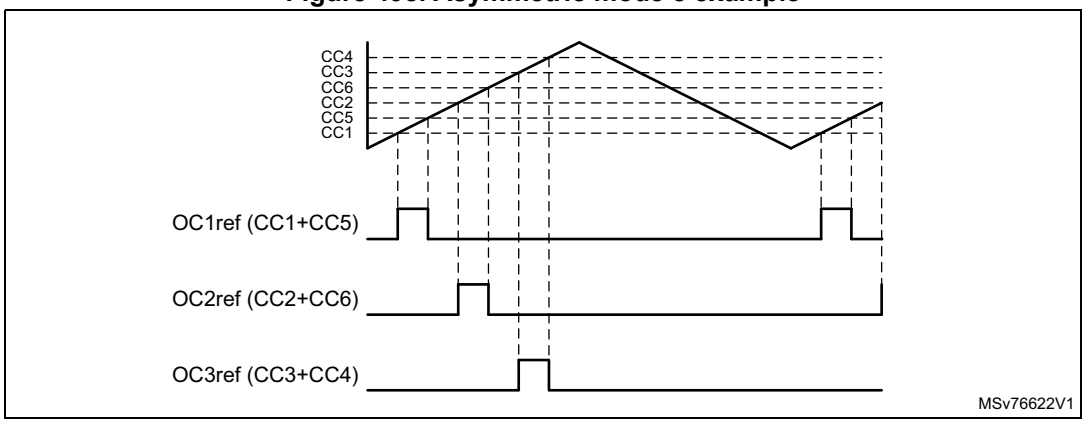
- Channel 1 is configured in asymmetric PWM mode 4
- Channel 2 is configured in asymmetric PWM mode 5
- Channel 3 is configured in regular PWM mode 2
- Compare 4 and 7 are available for ADC triggering purpose

Figure 497. Asymmetric mode example for single-shunt current sensing



The *Figure 498* represents an example of the waveforms that can be generated with asymmetric modes 7 to 10.

Figure 498. Asymmetric mode 8 example



41.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs, as summarized in the below:

Table 465. Combined PWM mode overview

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	oc5refc	oc6refc	oc7refc	PWM mode(1)
Combined PWM mode 1	01100	oc1ref OR oc2ref		oc3ref OR oc4ref		oc5ref OR oc6ref		-	1
Combined PWM mode 2	01101	oc1ref AND oc2ref		oc3ref AND oc4ref		oc5ref AND oc6ref		-	2
Combined PWM mode 3	10000	oc1ref OR oc5ref	oc2ref OR oc6ref	oc3ref OR oc5ref	oc4ref OR oc6ref	-	-	-	1
Combined PWM mode 4	10001	oc1ref AND oc5ref	oc2ref AND oc6ref	oc3ref AND oc5ref	oc4ref AND oc6ref	-	-	-	2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value.

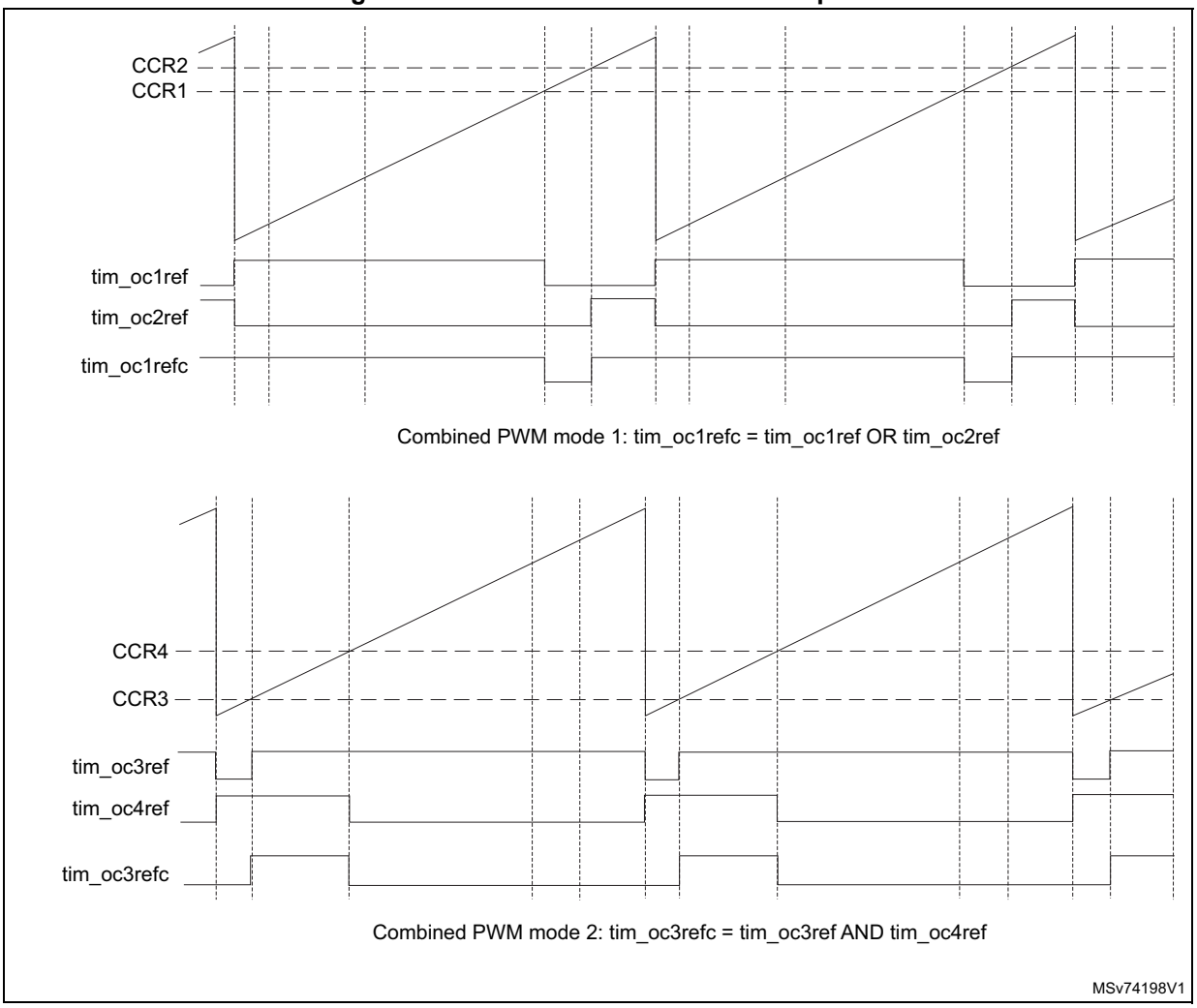
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in PWM mode 1 and the other in PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

Figure 499 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

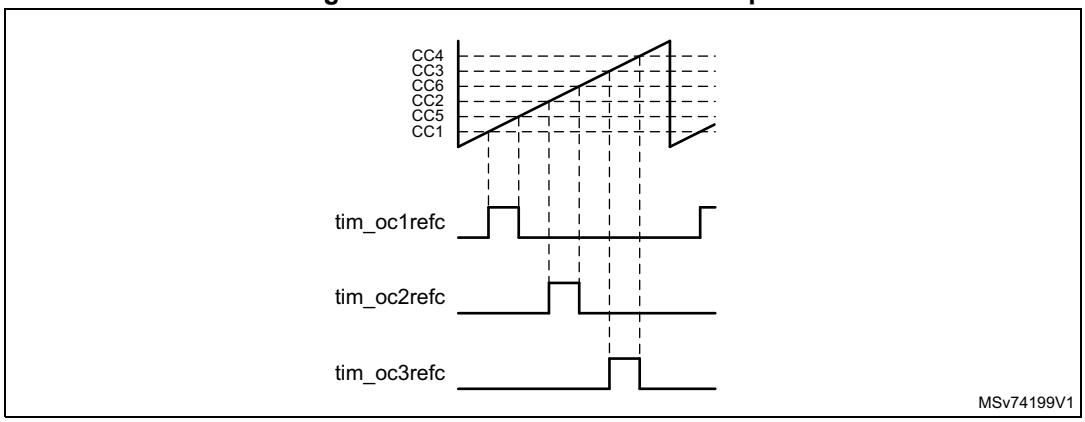
Figure 499. Combined PWM mode examples



The *Figure 500* represents an example suitable for a 3-phase interleaved converter requiring independent control of duty cycle and phase-shift. This is done using Combined PWM mode 2 and 4:

- Channel 1 is configured in Combined PWM mode 4
- Channel 2 is configured in Combined PWM mode 4
- Channel 3 is configured in Combined PWM mode 2

Figure 500. Combined PWM 3 example



41.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal.

The 4 bits GC5C[4:1] in the TIM_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is the logical AND of tim_oc1ref and tim_oc5ref
- If GC5C2 is set, tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
- If GC5C3 is set, tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
- If GC5C4 is set, tim_oc4refc is the logical AND of tim_oc4ref and tim_oc5ref

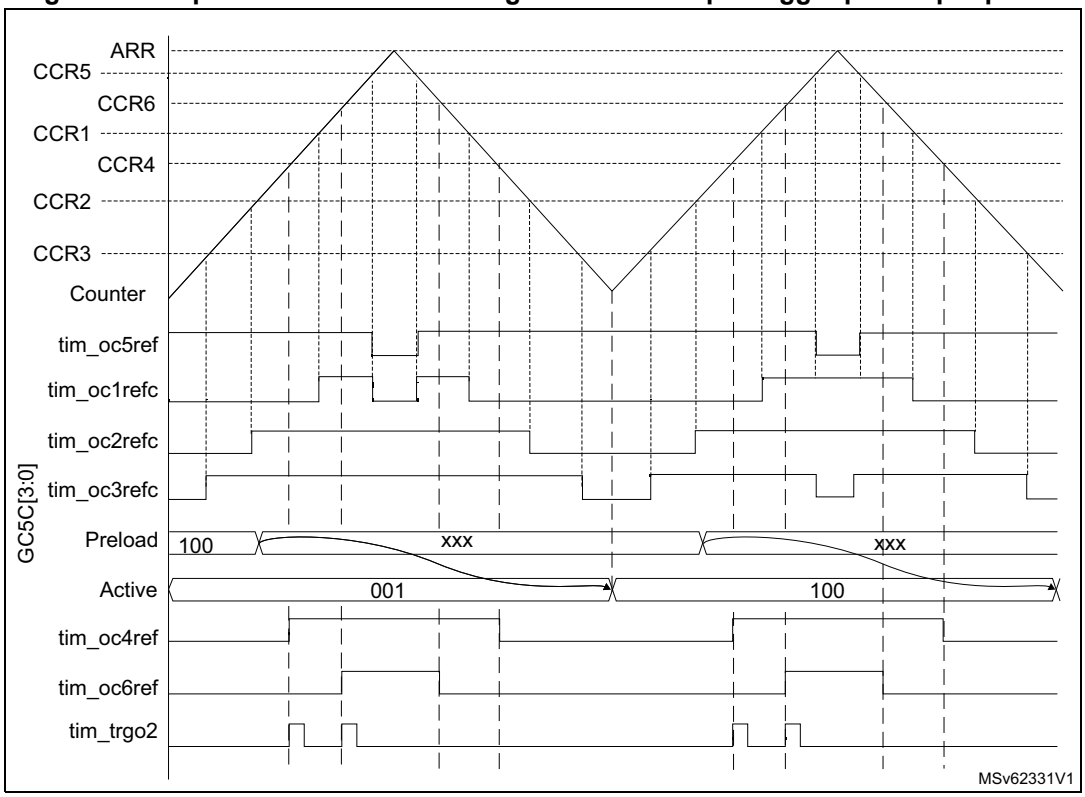
It is possible also to select an OR fonction with GC5C1O, GC5C2O, GC5C3O and GC5C4O bits, as following:

- If GC5C1O is set, tim_oc1refc is the logical OR of tim_oc1ref and tim_oc5ref
- If GC5C2O is set, tim_oc2refc is the logical OR of tim_oc2ref and tim_oc5ref
- If GC5C3O is set, tim_oc3refc is the logical OR of tim_oc3ref and tim_oc5ref
- If GC5C4O is set, tim_oc4refc is the logical OR of tim_oc4ref and tim_oc5ref

If both GC5Cx and GC5CxO bits are set, GC5Cx bit overrides GC5CxO programming and tim_ocxrefc is equal to tim_ocxref AND tim_oc5ref.

Combined 3-phase PWM mode can be selected independently on channels 1 to 4 by setting at least one of the bits GC5C[4:1] or GC5CO[4:1].

Figure 501. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 41.3.36: ADC triggers* for more details.

41.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8/TIM20) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or delays due to power switches).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIM_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIM_CCER register and the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIM_BDTR and TIM_CR2 registers. Refer to *Table 473: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

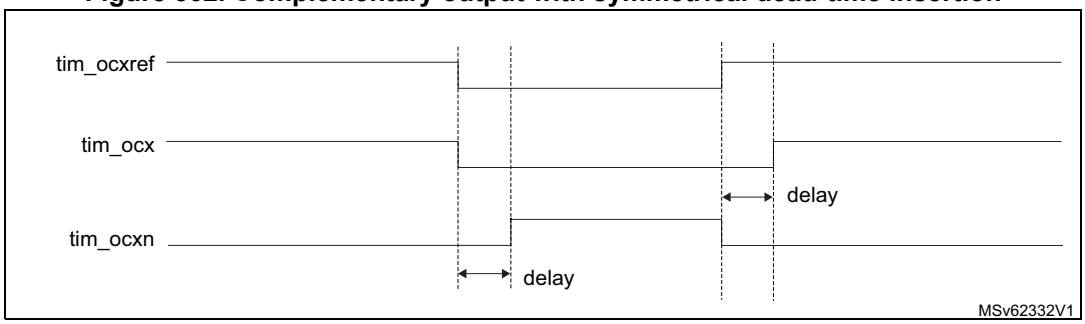
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 502. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIM_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 503*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIM_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIM_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIM_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 503. Asymmetrical deadtime

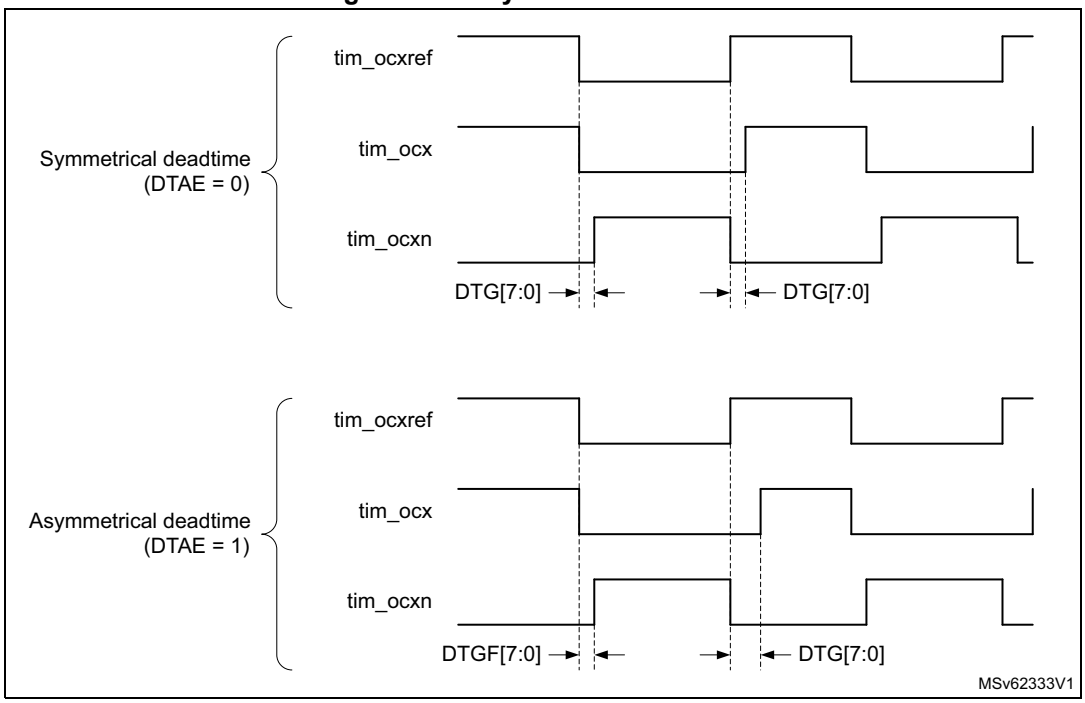


Figure 504. Dead-time waveforms with delay greater than the negative pulse

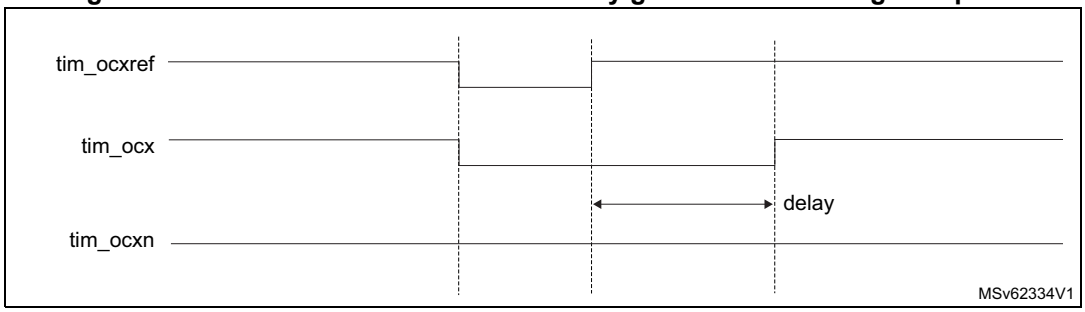
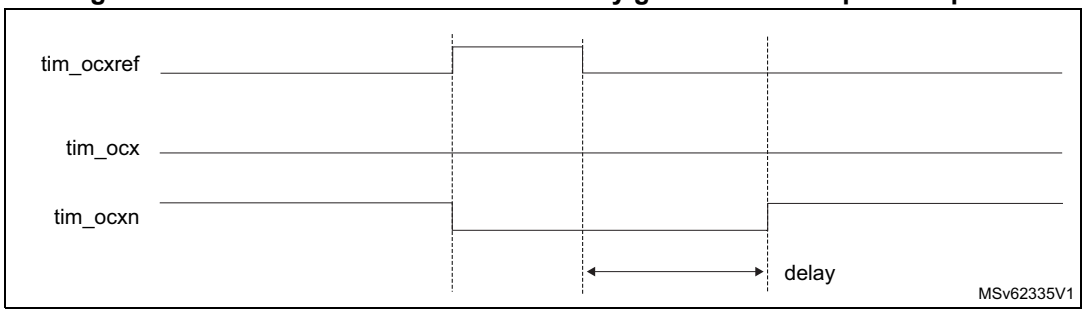


Figure 505. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIM_BDTR register. Refer to *Section 41.6.20: TIM break and dead-time register (TIM_BDTR)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIM_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

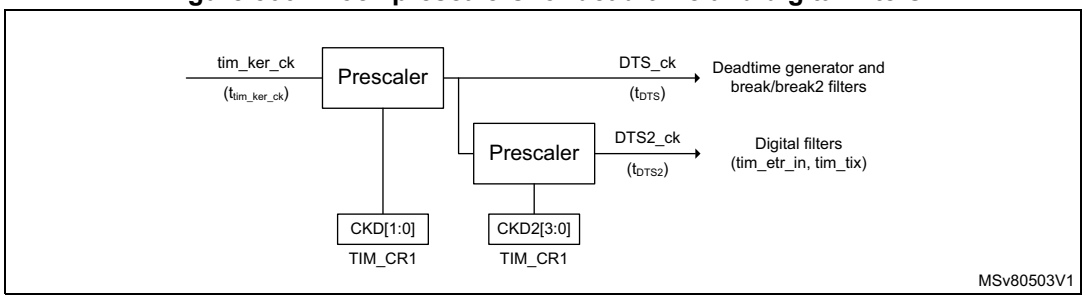
Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

41.3.18 Clock dividers for dead-time generators and digital filters

The dead-time generators and the digital filters (for tim_etr_in, tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the deadtime period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend solely the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 506. Clock prescalers for dead-time and digital filters



41.3.19 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIM_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIM_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- The OISx and OISxN bits in the TIM_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 473: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIM_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIM_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)

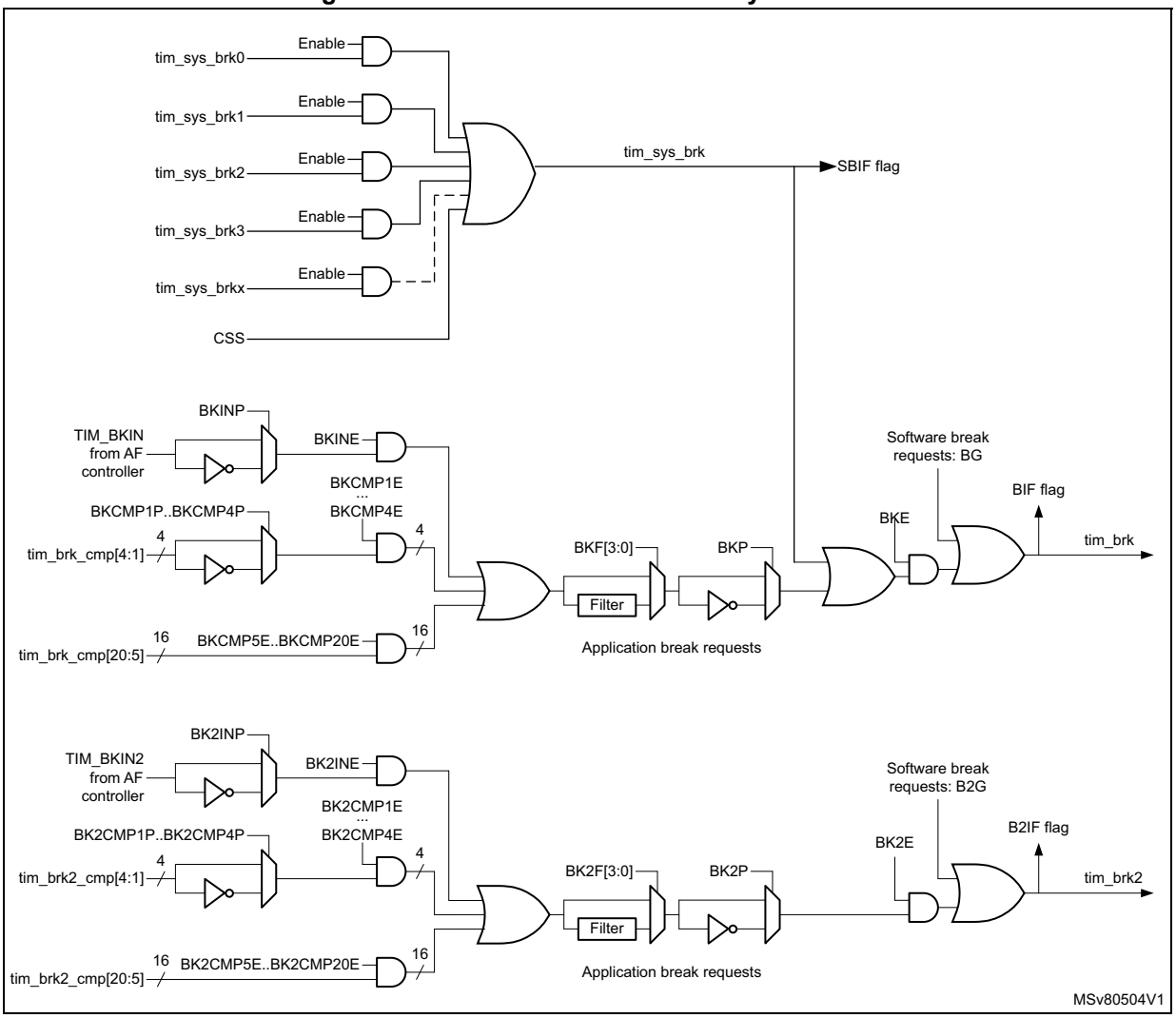
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIM_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIM_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 507* below.

Figure 507. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIM_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flags (B2GF, BGF, SBIF, BIF, and B2IF bits in the TIM_SR register) are set. An interrupt is generated if the BIE bit in the TIM_DIER register is set.
- An output disable status flag (ODS in the TIM_SR register) indicates the status of the outputs:
  - ODS = 0: Idle state. Break was triggered (or MOE was written to 0) and OIS states are forced on tim_ocx and tim_ocxn outputs
  - ODS = 1: Disabled state. Break2 was triggered, tim_ocx and tim_ocxn outputs are inactive
- If the AOE bit in the TIM_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

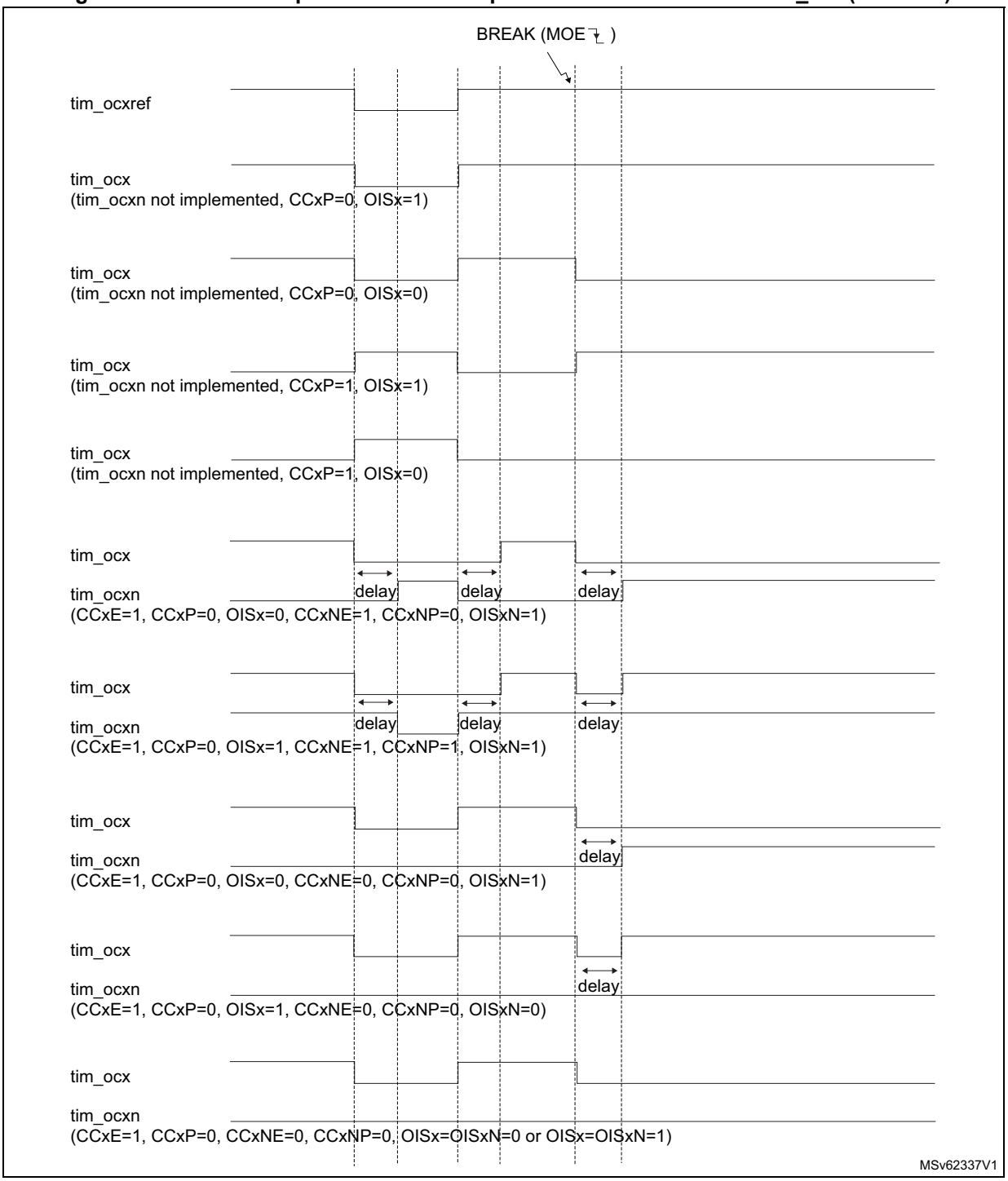
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIM_CR2 register.

The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The application can choose from three levels of protection selected by the LOCK bits in the TIM_BDTR register. Refer to *Section 41.6.20: TIM break and dead-time register (TIM_BDTR)*. The LOCK bits can be written only once after an MCU reset.

Figure 508 shows an example of behavior of the outputs in response to a break.

Figure 508. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 466*.

The output disable status flag (ODS in the TIM_SR register) indicates the status of the outputs after a break event and is useful to distinguish break and break2 safe states.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 466. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 509 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIM_CCER register).

Figure 509. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

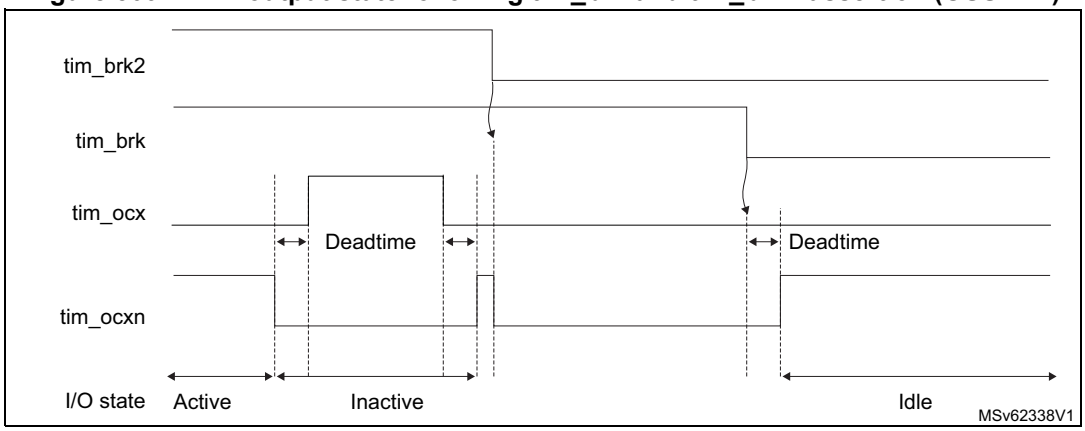
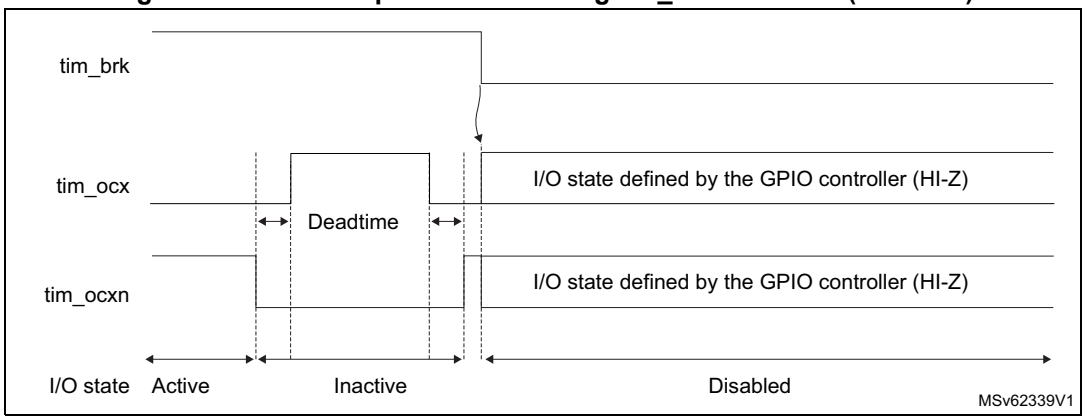


Figure 510. PWM output state following tim_brk assertion (OSSI = 0)



41.3.20 Delayed break

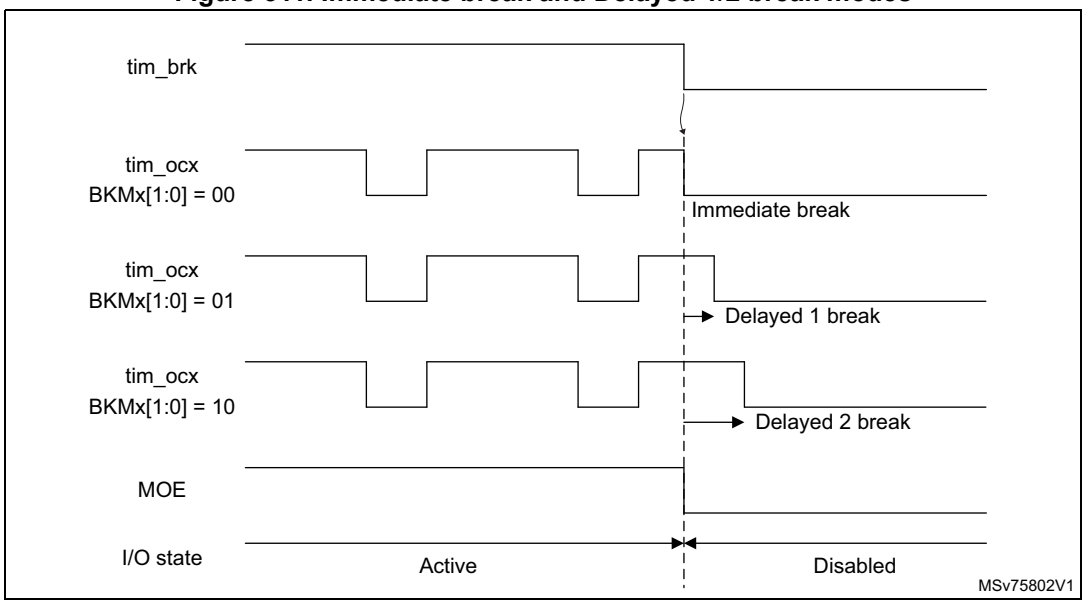
In Neutral-Point Clamped (NPC) multilevel converter topology, a break strategy with immediate outputs shut-down is not suitable, as it may damage the power switches with excessive voltage levels.

The delayed break features provide 2 options delaying the response to a break event, and implement a safe shut-down sequence.

BKMx[1:0] and BKMxN[1:0] bitfields in TIM_MPR1 provides 3 options for each complementary outputs (tim_oc1/tim_oc1n to tim_oc4/tim_oc4n), as show on *Figure 511* below:

- BKM[1:0] = 00: the break response is immediate
- BKM[1:0] = 01: Delayed 1 break. The break response is delayed with a delay defined in the DBK1[7:0] bitfield of the TIM_MPR2 register
- BKM[1:0] = 10: Delayed 2 break. The break response is delayed with a delay defined in the DBK2[7:0] bitfield of the TIM_MPR2 register

Figure 511. Immediate break and Delayed 1/2 break modes



The delayed break mode is triggered when:

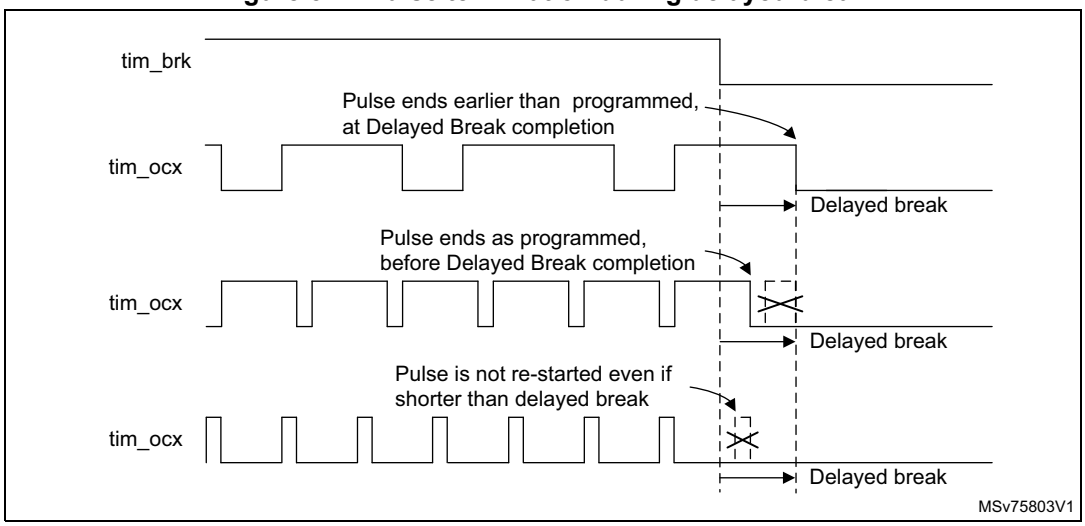
- A break event occurs
- The MOE bit is reset
- When the microcontroller enters debug mode with DBG_TIM_STOP=1

Warning: The break2 feature must be disabled when the delayed mode is enabled.

When delayed break mode is triggered, the MOE bit is immediately cleared, and cannot be set during the delayed break duration. In bidirectional break mode, the break output is reset immediately to indicate the delayed break start to external components. Once the longest of the 2 delays is over, the deadtime generator is enabled to prevent any deadtime violation if OISx/OISxN bits are set.

If the tim_ocx/tim_ocxn signal is reset before the end of the delayed break (due to a compare match in PWM mode, typically), the signal is not maintained during the full delayed break period, to maintain the duty cycle, as shown on *Figure 512* below. Still, the Delayed break circuitry guarantees that the tim_ocx/tim_ocxn signals cannot be set again once the break event occurred.

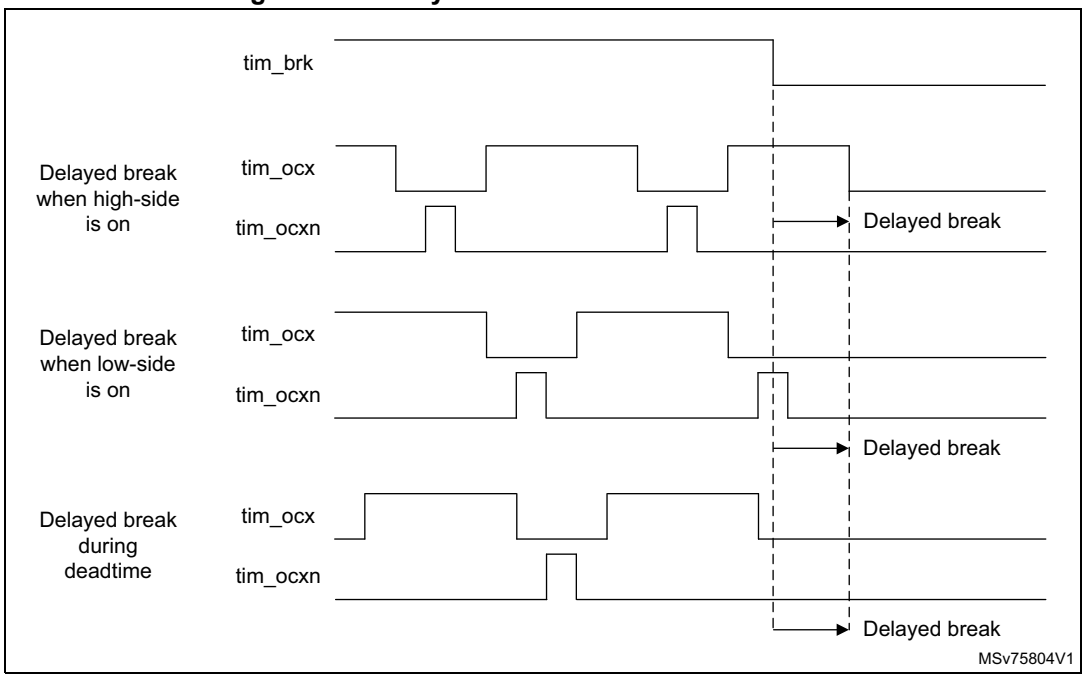
Figure 512. Pulse termination during delayed break



The Figure 563 depicts the behavior when the deadtime generator is enabled, for the 3 cases:

– Delayed break triggered when tim_ocx is set
– Delayed break triggered when tim_ocxn is set
– Delayed break triggered during deadtime interval

Figure 513. Delayed break with deadtime insertion



The delay programming is identical to deadtime generator's, with 2 user-programmable durations in the TIM_MPR2 register: DBK1[7:0] bitfield for the Delayed 1 break and DBK2[7:0] bitfield for the Delayed 2 break.

DBK1[7:0] and DBK2[7:0] bitfields are preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

DBK1[7:0] and DBK2[7:0] bitfields are partially preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

The bits coding for t_{DTG} selection are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBKx[6:0] if DBKx[7] = 0
- DBKx[5:0] if DBKx[7:6] = 10
- DBKx[4:0] if DBKx[7:5] = 110 or 111

The BKMx[1:0] and BKMxN[1:0] bits are static, they can only be modified when the counter is stopped (CEN=0 in TIM_CR1).

The content of the TIM_MPR1 and TIM_MPR2 register can be write protected using the LOCK[1:0] bits in the TIM_BDTR register.

41.3.21 Bidirectional break inputs

The TIM1/TIM8/TIM20 feature bidirectional break I/Os, as represented on *Figure 514*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIM_BDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIM_BDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN and TIM_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 467*).

Table 467. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

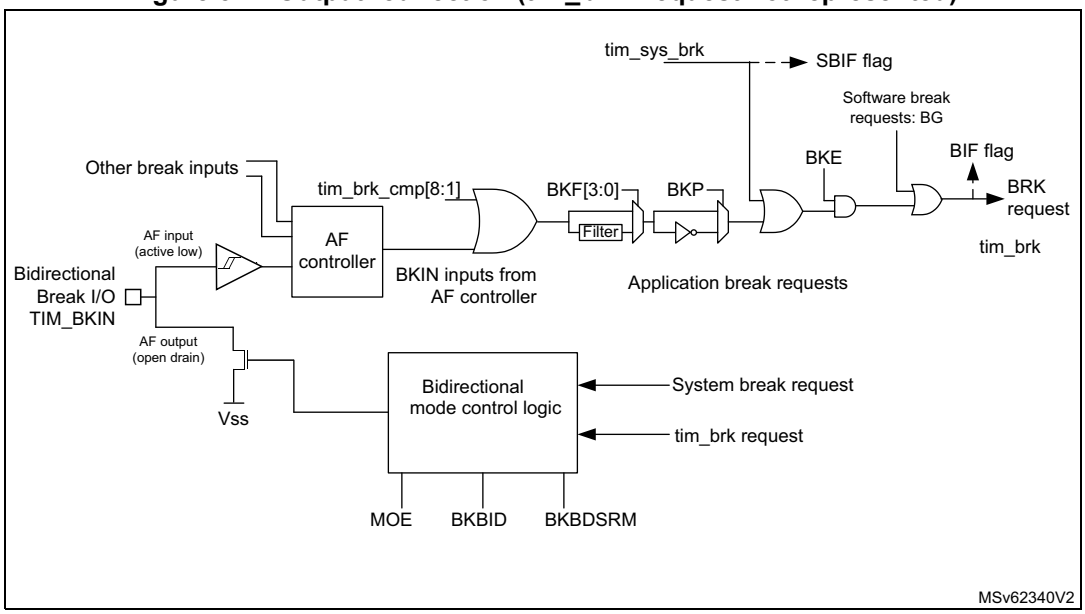
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 514. Output redirection (tim_brk2 request not represented)



41.3.22 Output idle state override

It is possible to override the outputs states when they are disabled (ie MOE bit reset in TIM_BDTR). This feature can be used to apply specific patterns prior to start PWM operations, for instance to refresh bootstrap capacitors of power switch gate drivers.

The OOC control bit in the TIM_OOR register acts as following:

- OOC= 0: 0: tim_ocx and tim_ocxn outputs are in the default idle state
- OOC=1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits in the TIM_OOR register

The OOC bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared by hardware when the MOE bit is set, and the deadtime generator ensures there's no deadtime violation on complementary outputs.

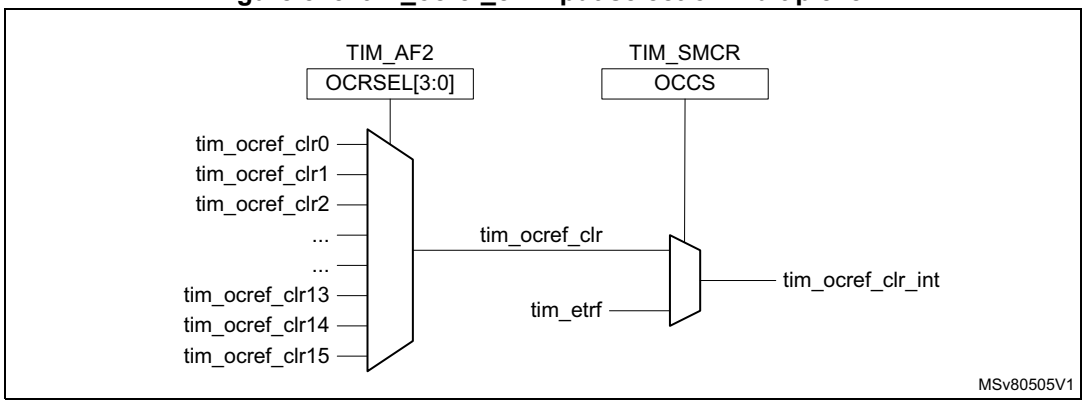
The OOSx/OOSxN bits in the TIM_OOR register cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

41.3.23 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It must not be used in Forced mode. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etr_in after the filter) by configuring the OCCS bit in the TIM_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[3:0] bitfield in the TIM_AF2 register, as shown on the *Figure 515* below. Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for a list of sources available in the product.

Figure 515. tim_ocref_clr input selection multiplexer

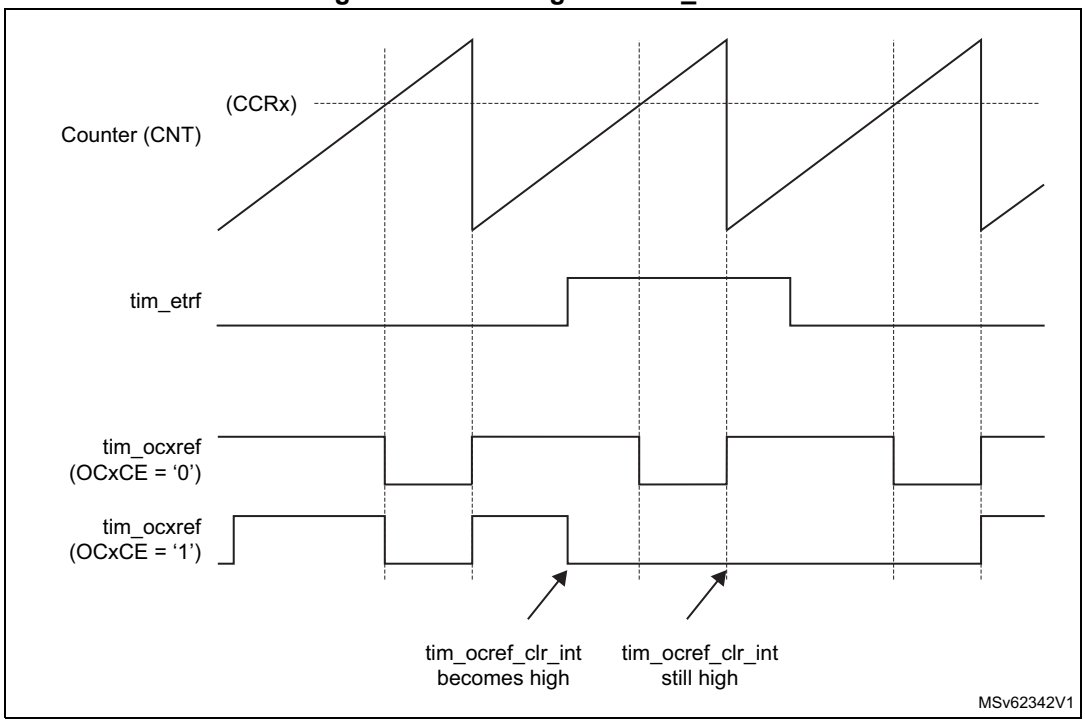


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIM_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIM_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 516 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 516. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to have the OCxCE bit enabled for the channel, the OCxCE bit of the complementary channel (e.g. OC2CE = X for oc2ref in *Figure 517*) is ignored in this case.

Figure 517 to *Figure 520* are describing the output waveforms for some use cases, when the ocref_clr_int signal is asserted.

Figure 517. Asymmetric PWM mode 1

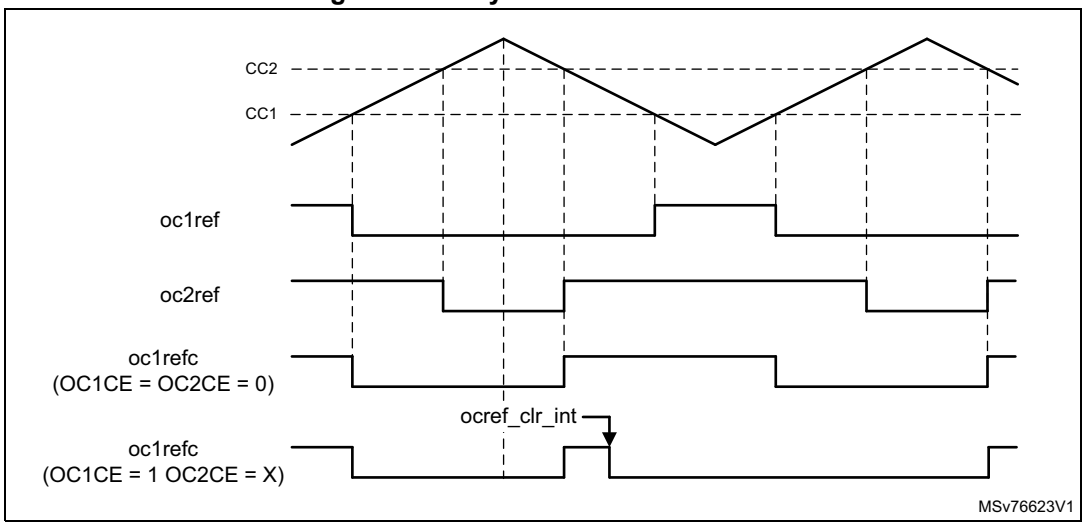


Figure 518. Asymmetric PWM mode 2

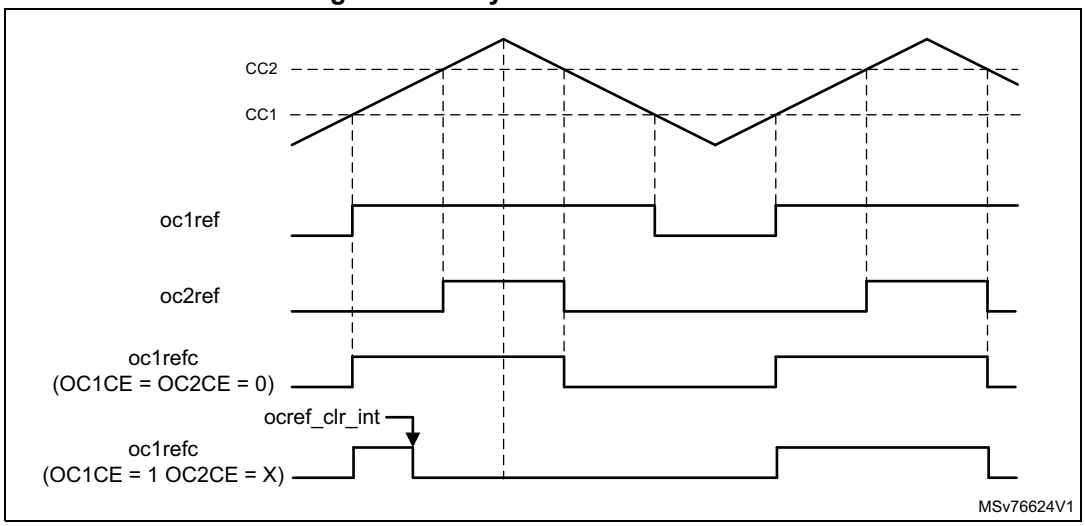


Figure 519. Asymmetric PWM mode 7 (OC1refc = oc1ref OR oc5ref)

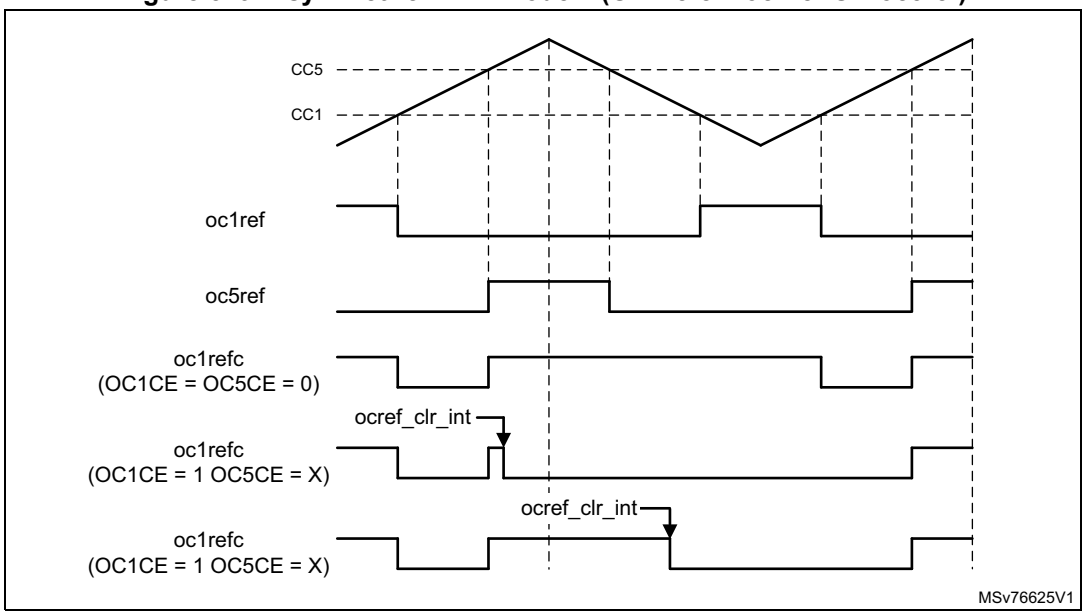
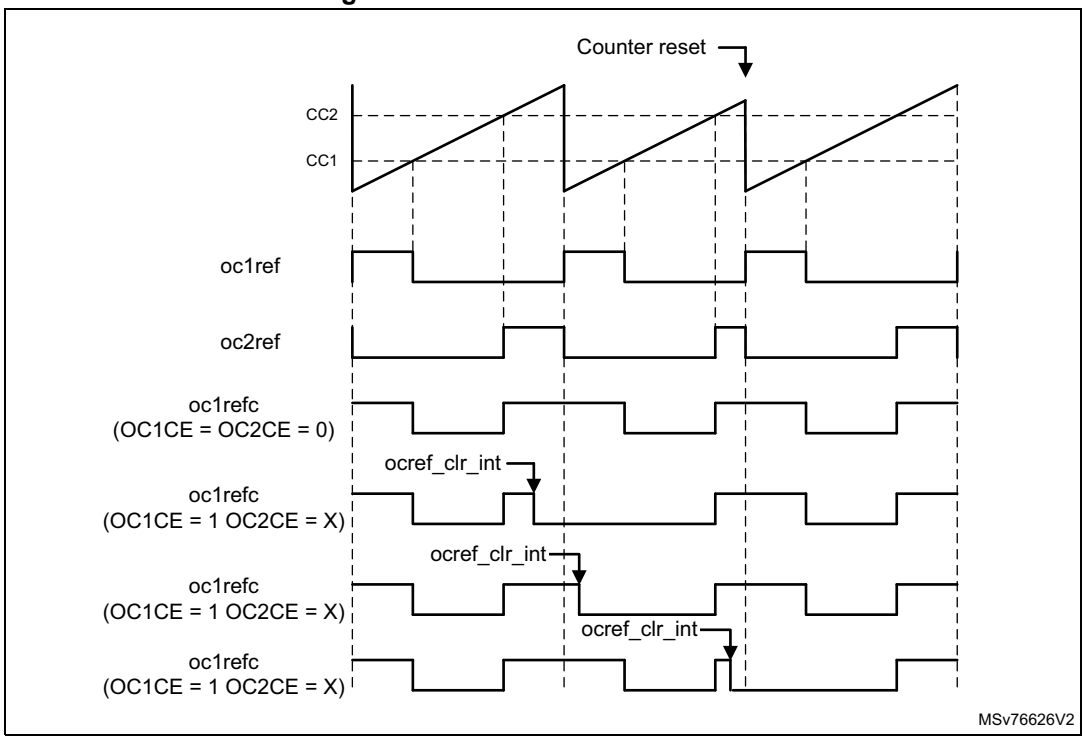


Figure 520. Combined PWM mode 1



Conditions of use

This feature is intended to support cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used for the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One pulse mode with PWM active on match (OPM=1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E, 0x10, 0x12, 0x14, 0x16, 0x18)
- Combined 3-phase modes (GC5Cx or GC5CxO = 1)
- Combined PWM mode 1 or 3 in center aligned mode (CMS[1:0] = 1,2,3 and OCxM[4:0] = 0x0C or 0x10)

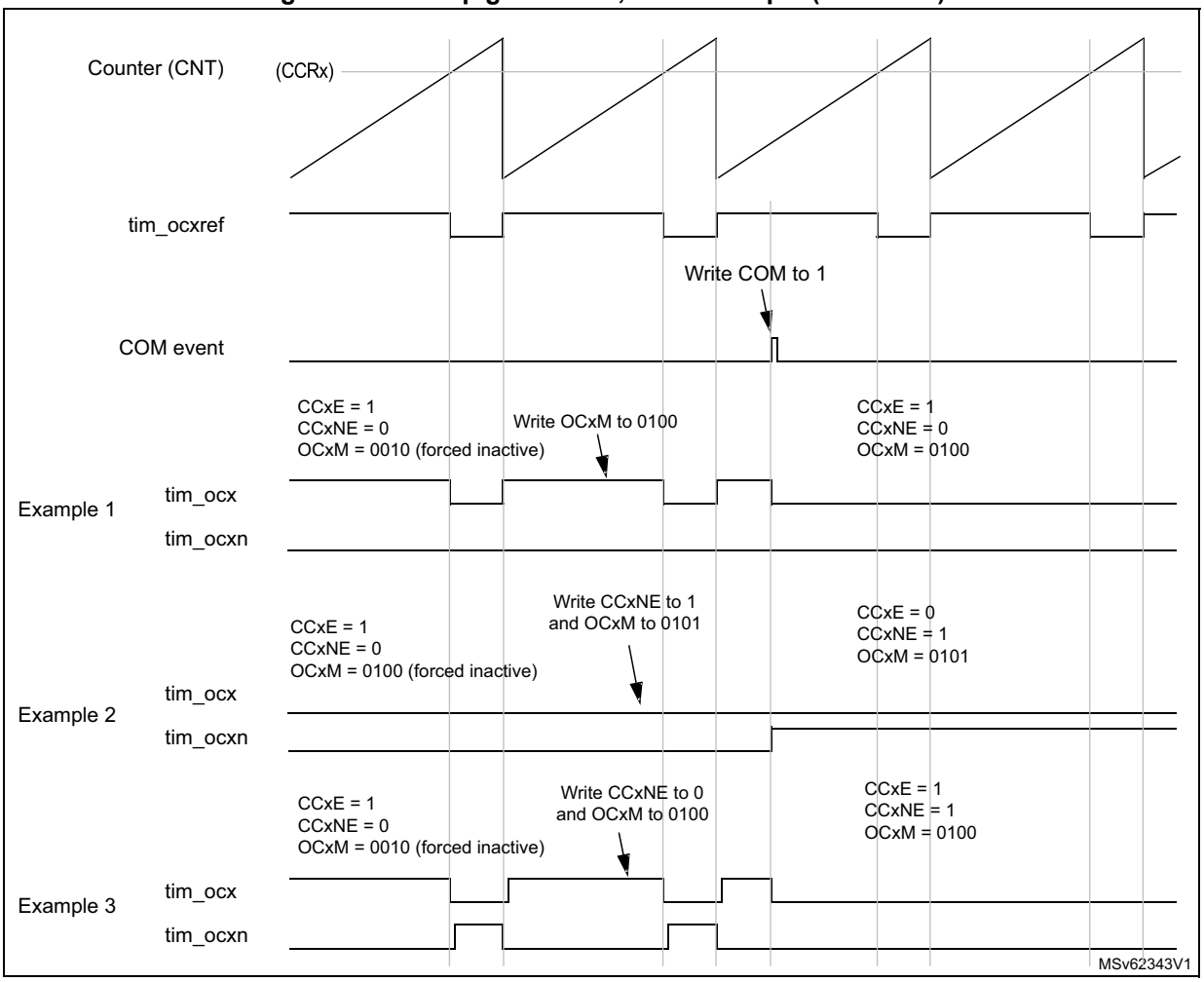
41.3.24 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIM_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIM_SR register), which can generate an interrupt (if the COMIE bit is set in the TIM_DIER register) or a DMA request (if the COMDE bit is set in the TIM_DIER register).

Figure 521 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 521. 6-step generation, COM example (OSSR = 1)



41.3.25 One-pulse mode

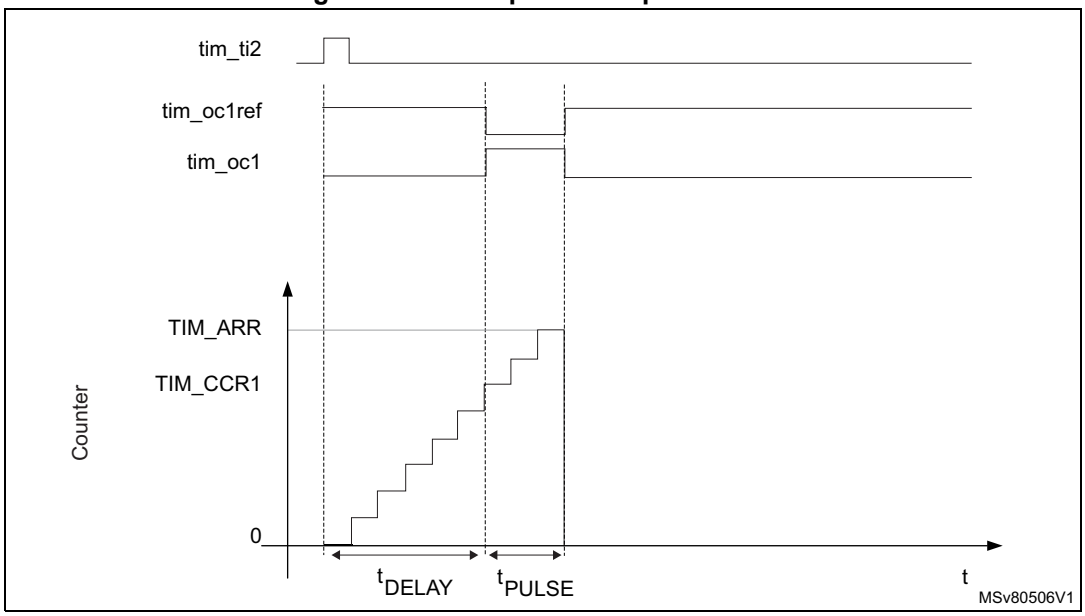
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 522. Example of one pulse mode.



In the following example, the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 00111 in TIM_CCMR1). Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIM_CR1 register must be low.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over

from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be achieved.

To output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

41.3.26 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one-pulse mode described in *Section 41.3.25*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

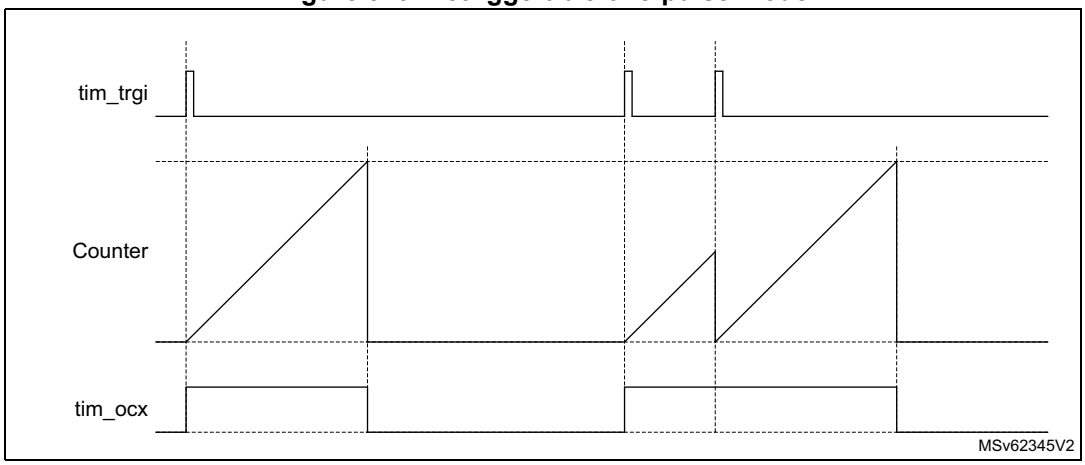
The timer must be in Slave mode, with the bits SMS[4:0] = 01000 (Combined Reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[4:0] and SMS[4:0] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 523. Retriggerable one-pulse mode



41.3.27 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 524*.

Figure 524. Pulse generator circuitry

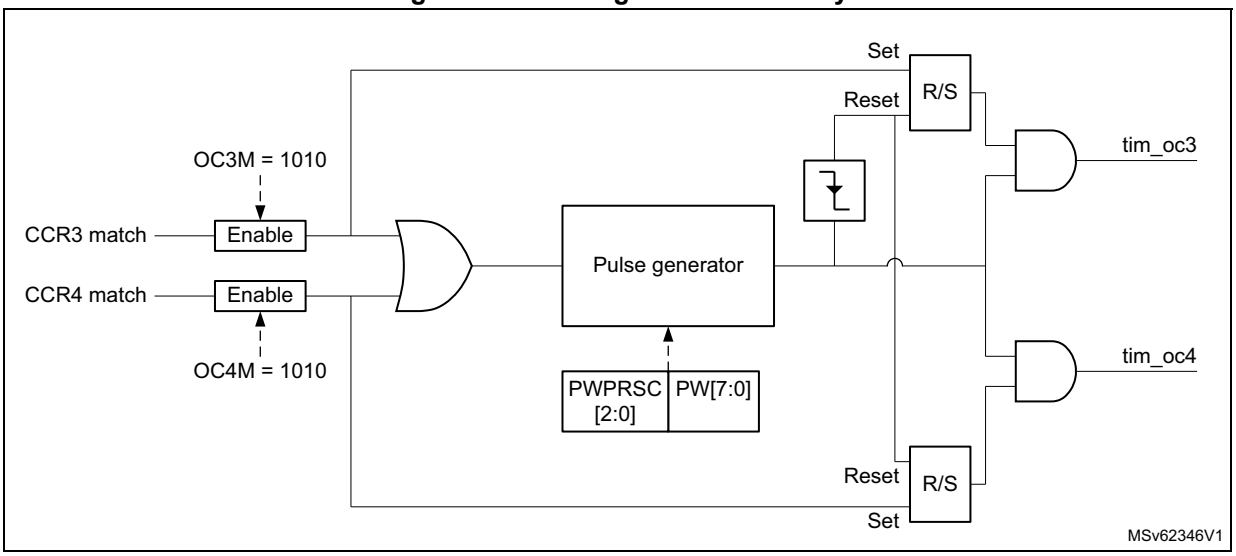
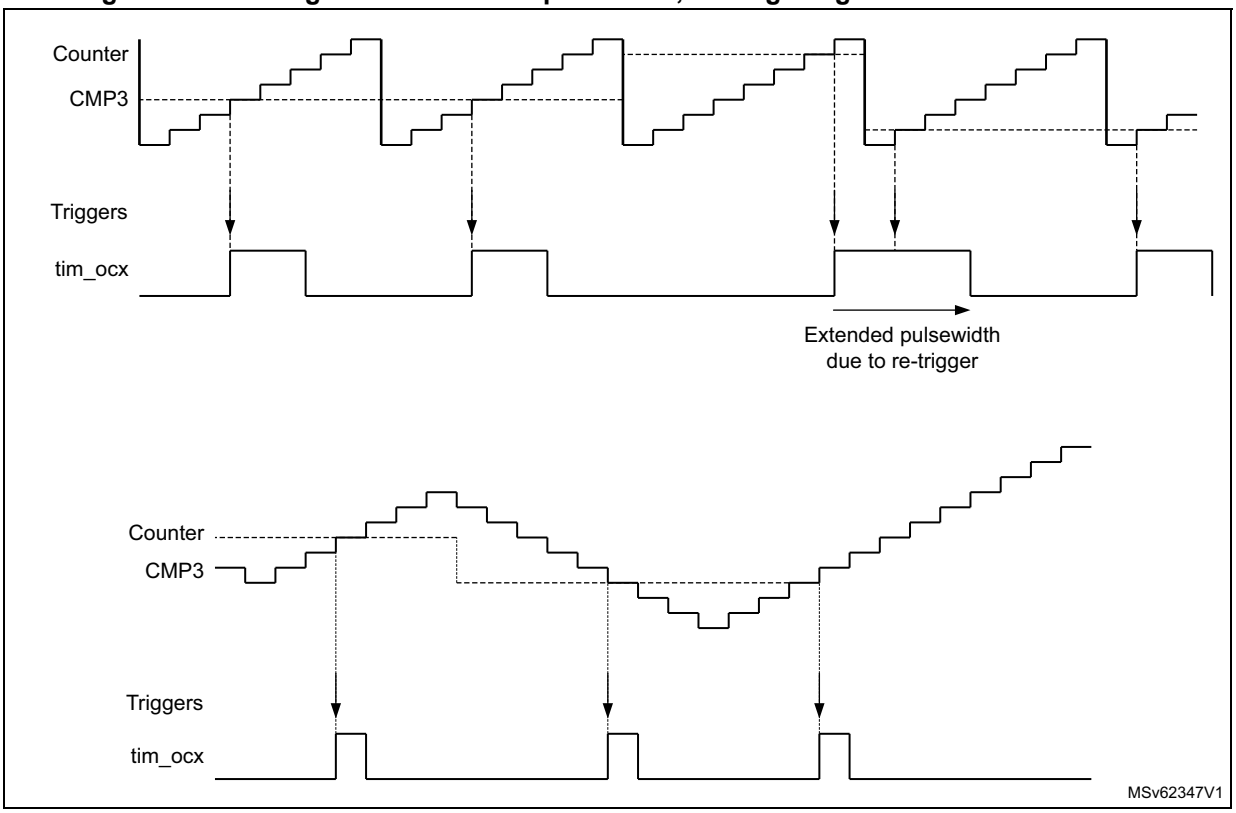


Figure 525 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 525. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIM_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

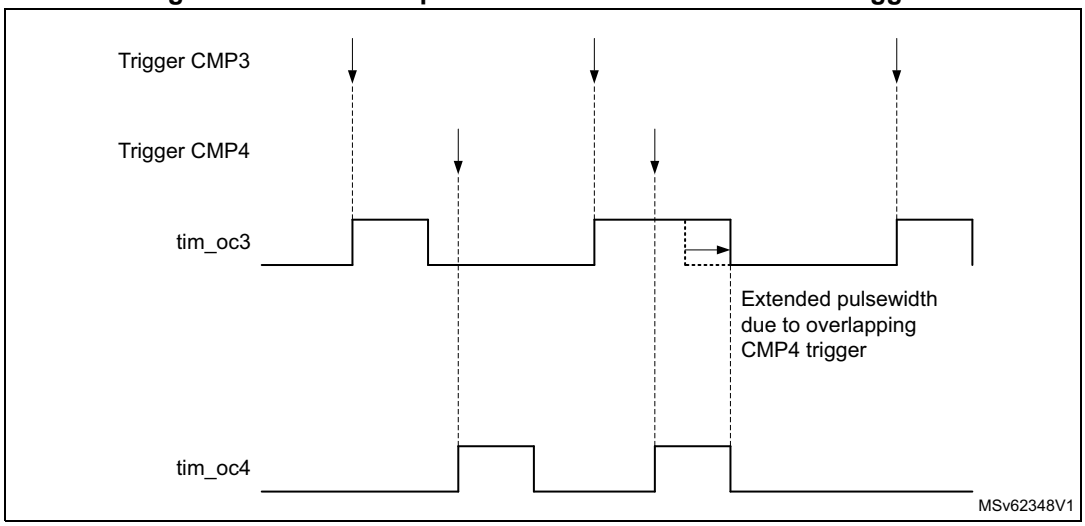
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 526).

Figure 526. Extended pulsewidth in case of concurrent triggers



41.3.28 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 0001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 468*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 468. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 527 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIM_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, Counter enabled).

Figure 527. Example of counter operation in encoder interface mode.

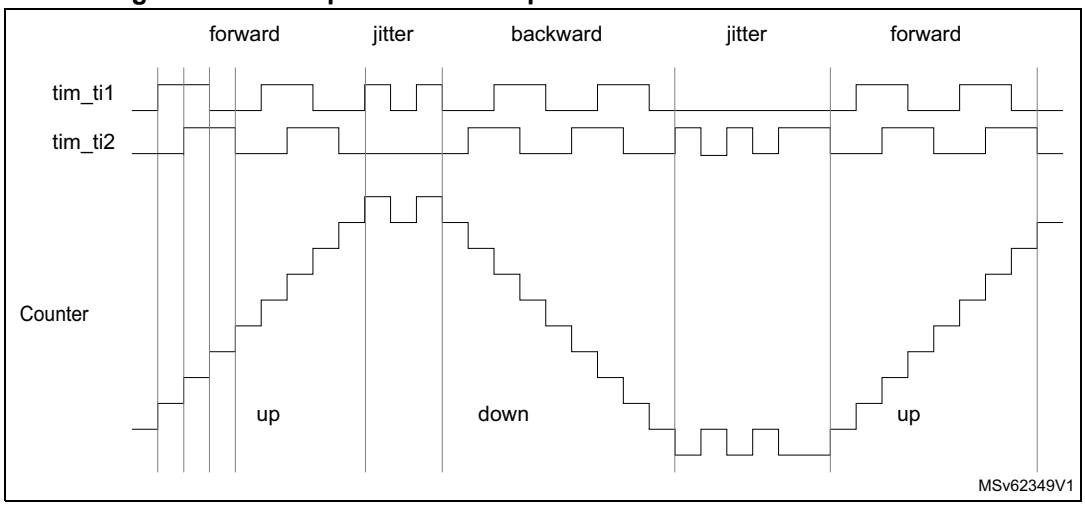


Figure 528 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 528. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

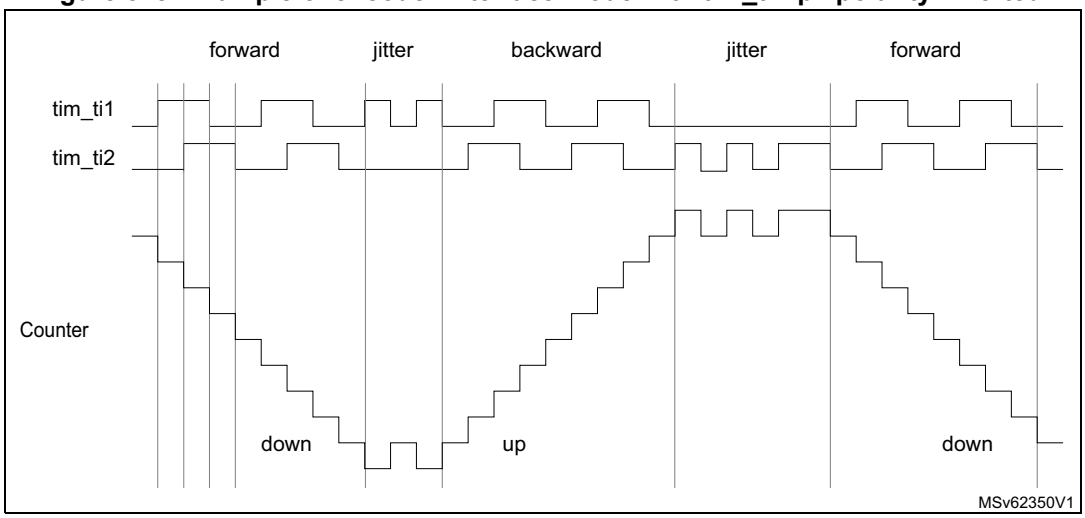
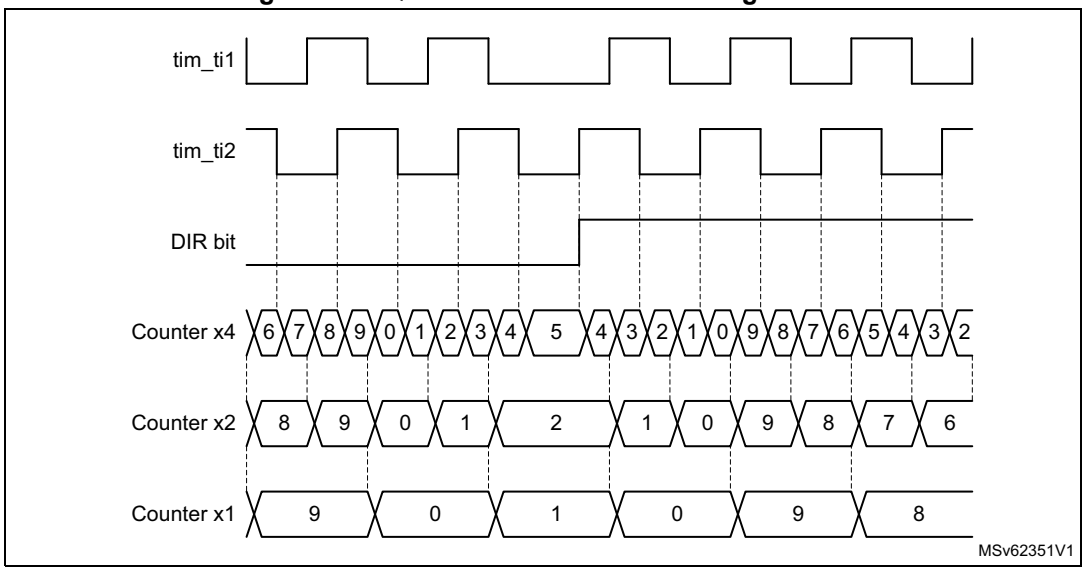


Figure 529 shows the timer counter value during a speed reversal, for various counting modes.

Figure 529. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

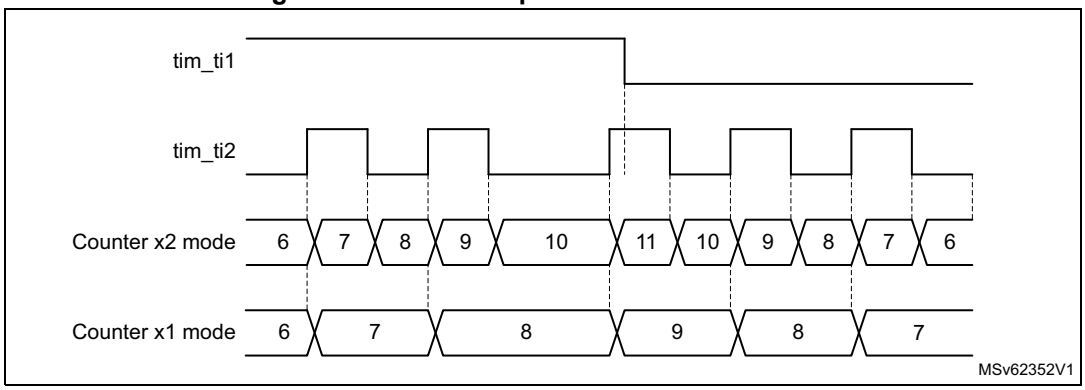
In the clock plus direction mode shown on *Figure 530*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 530. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 531*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 531*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 532*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 531*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 532*).

Figure 531. Directional clock encoder mode (CC1P = CC2P = 0)

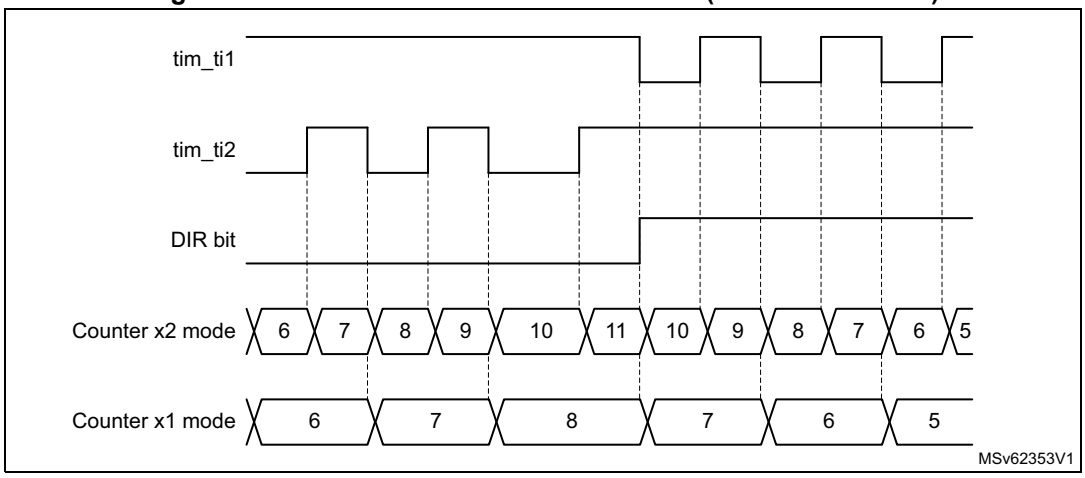


Figure 532. Directional clock encoder mode (CC1P = CC2P = 1)

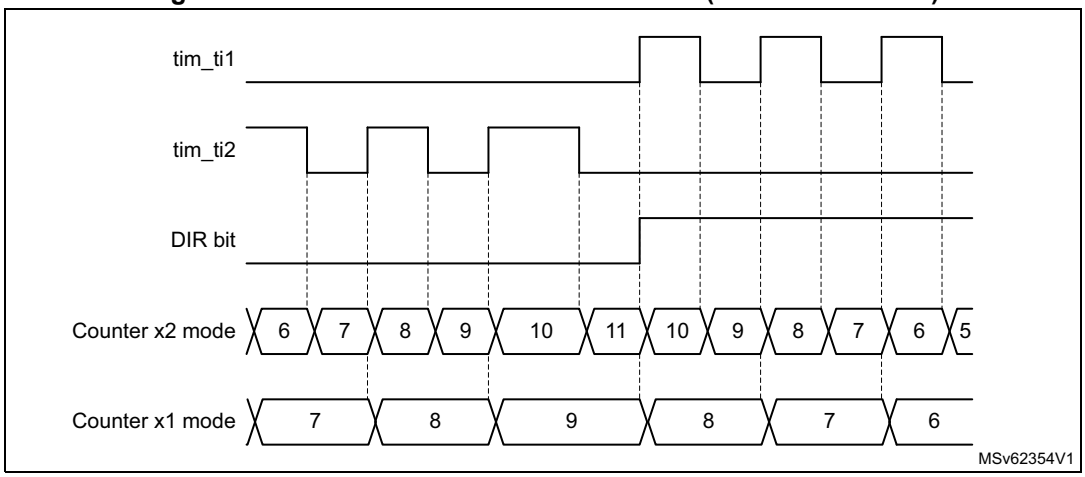


Table 469 here-below details how the directional clock mode operates, for any input transition.

Table 469. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

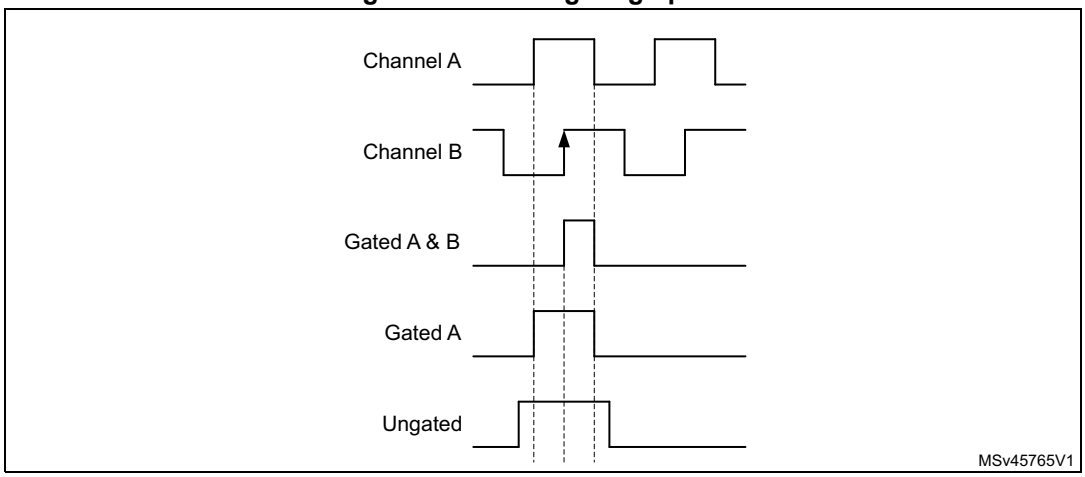
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIM_ECR register. The IE bit must be set only in encoder mode, when the SMS[4:0] bitfield has the following values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 533*:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

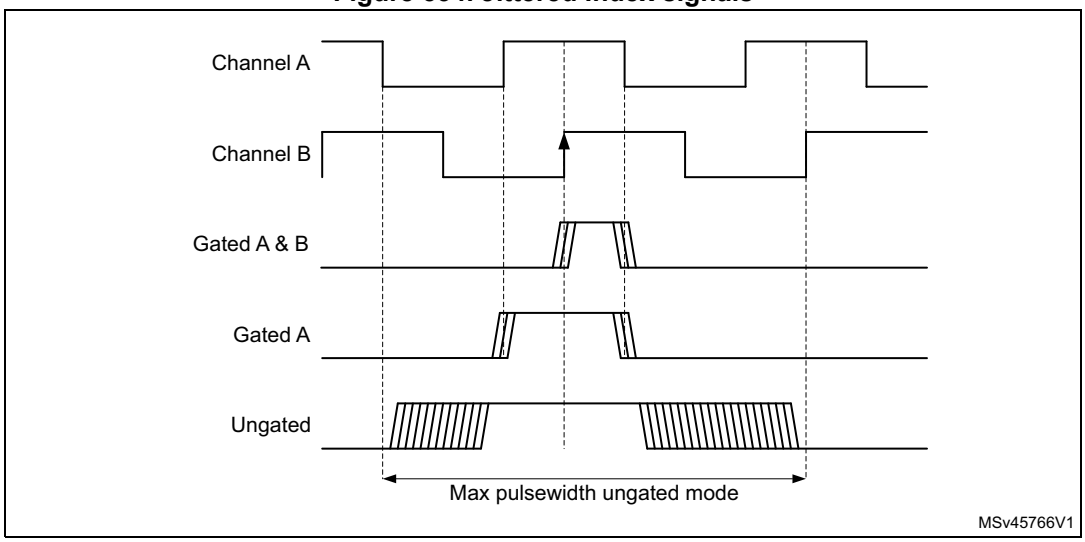
Figure 533. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 534*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 534. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and

channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIM_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIM_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 535* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 535. Index generation for IPOS[1:0] = 11

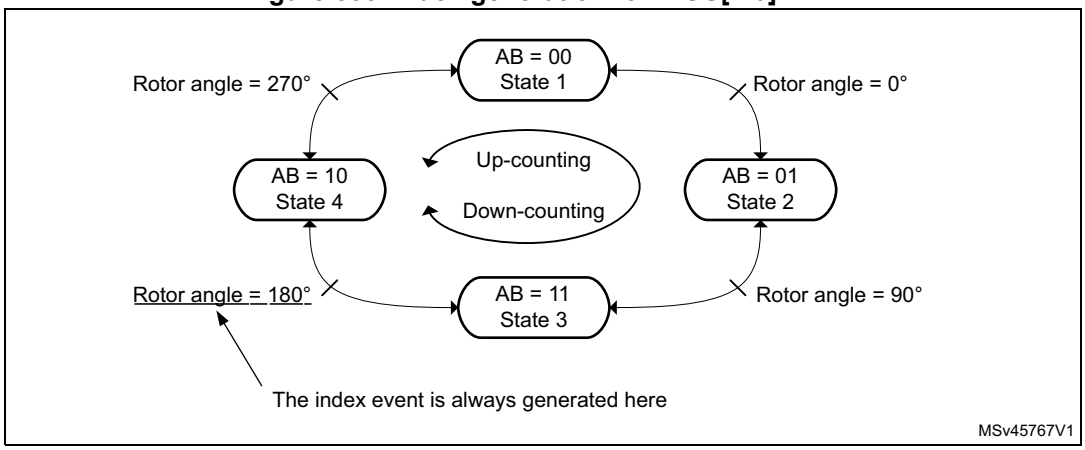


Figure 536 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIM_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 536. Counter reading with index gated on channel A (IPOS[1:0] = 11)

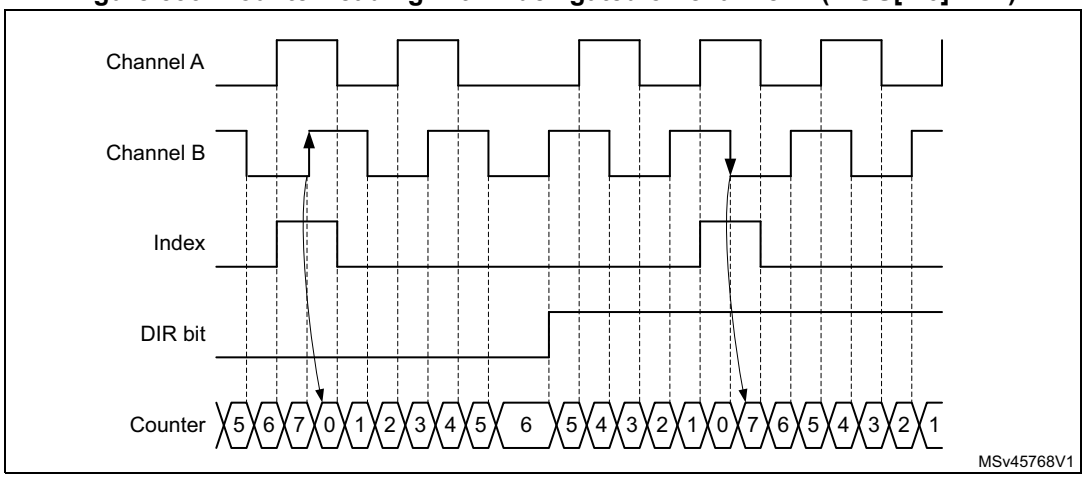


Figure 537. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 537. Counter reading with index ungated (IPOS[1:0] = 00)

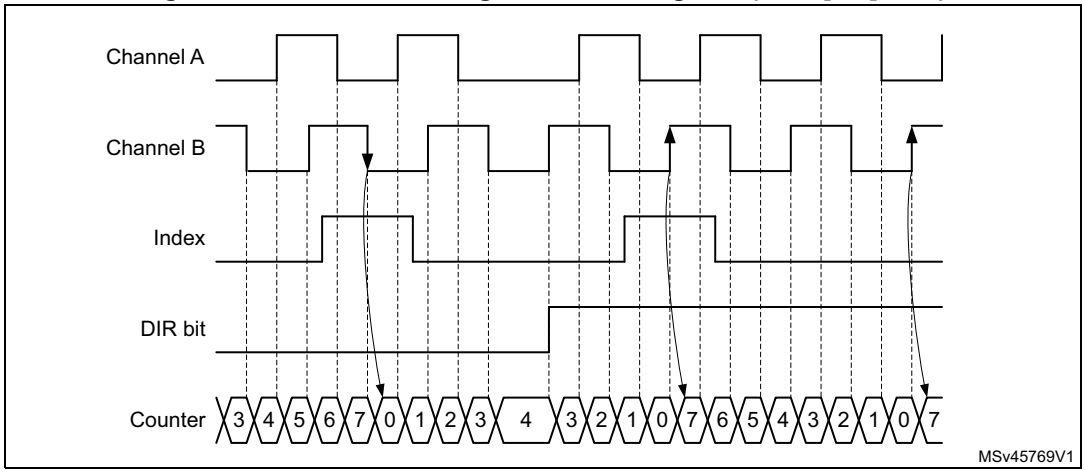


Figure 538. shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 538. Counter reading with index gated on channel A and B

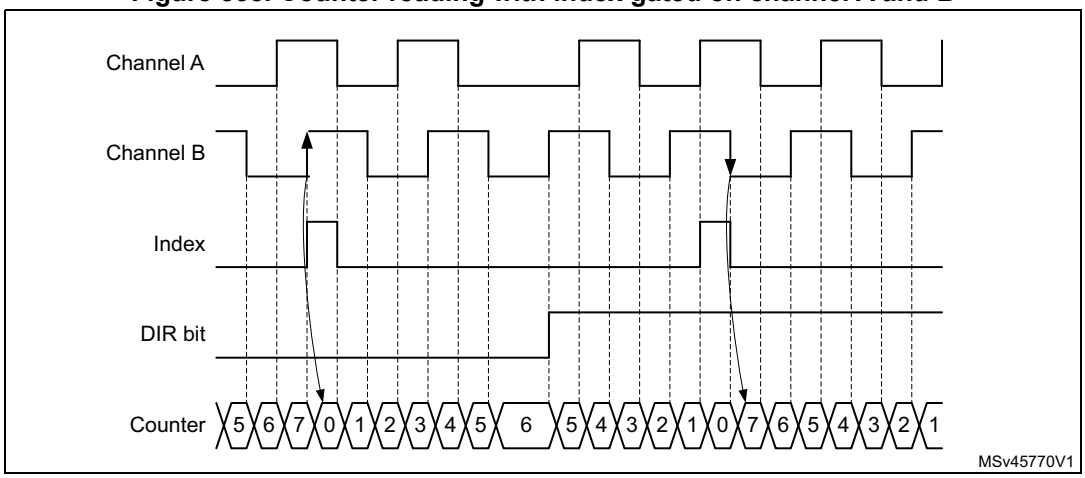


Figure 539 and *Figure 540* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 539. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

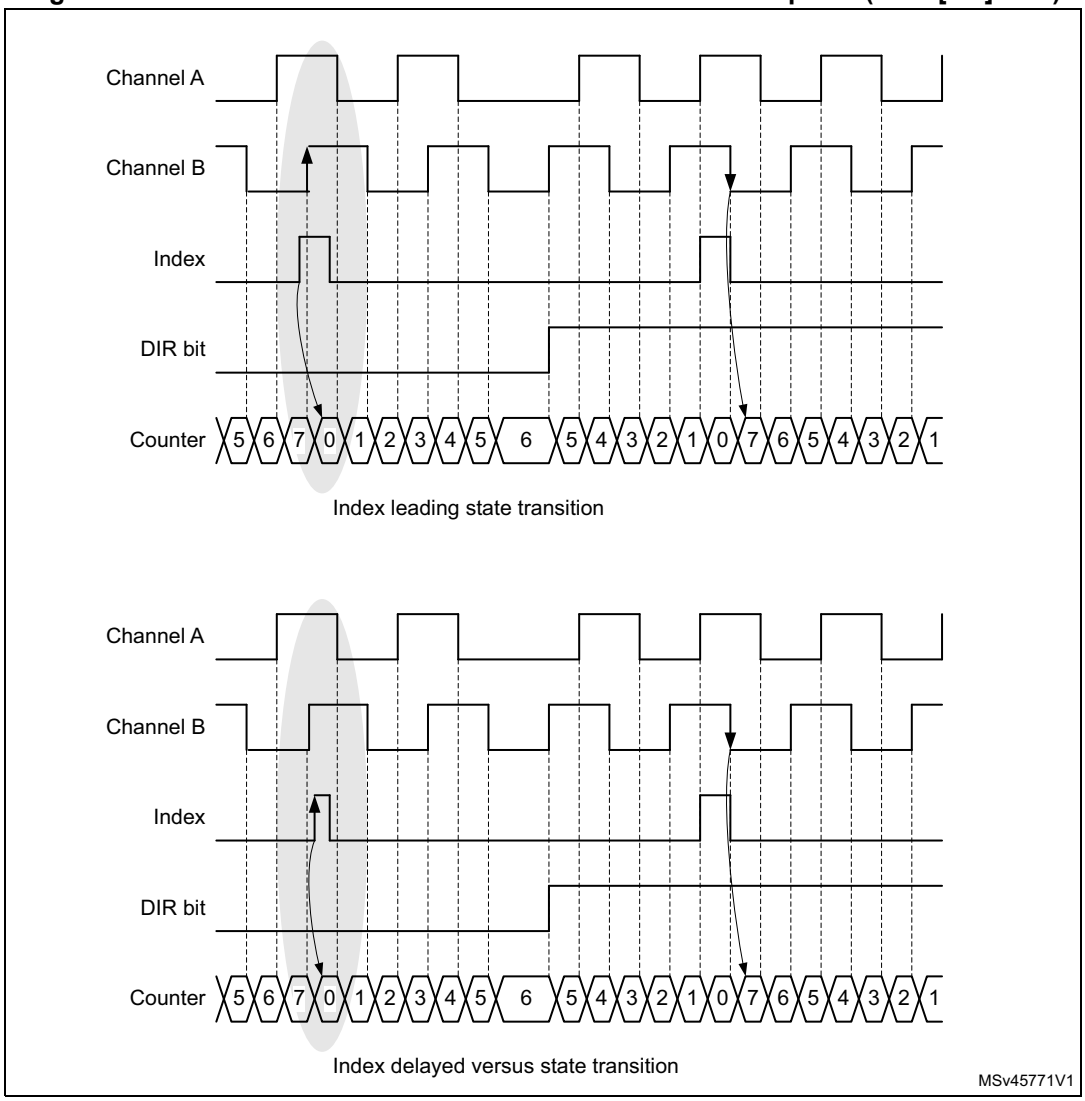


Figure 540. Counter reset Narrow index pulse (closer view, ARR = 0x07)

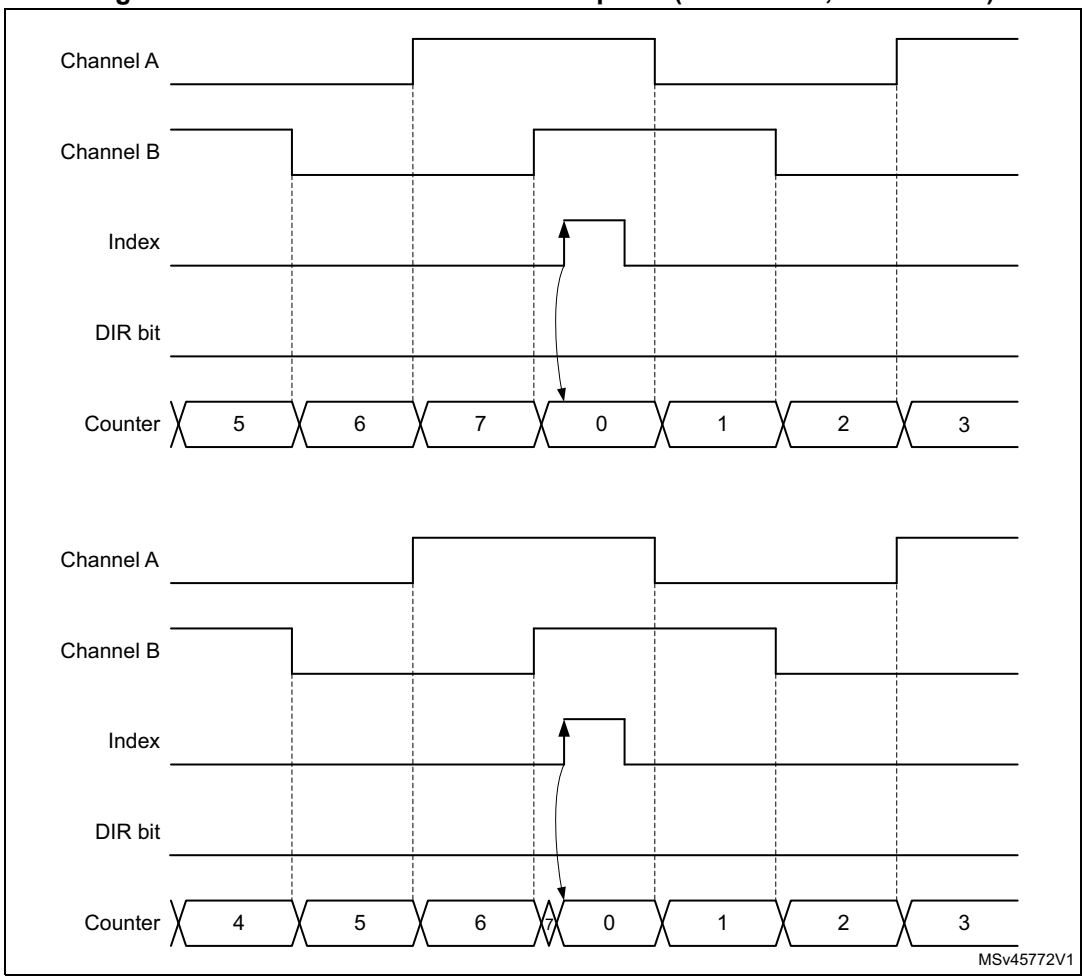
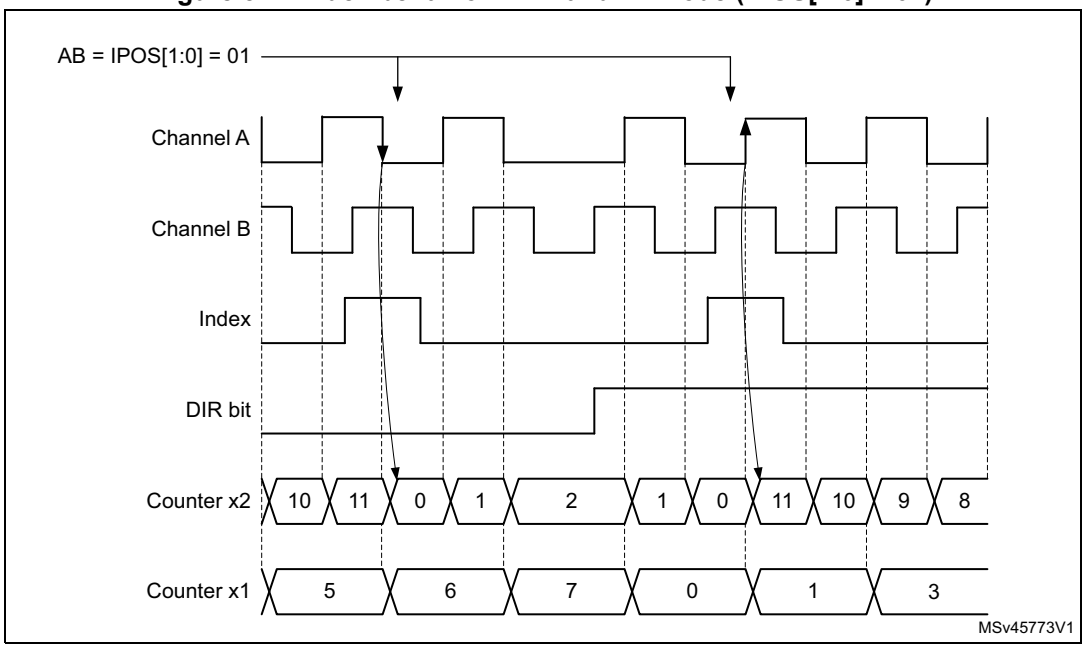


Figure 541 shows how the index is managed in x1 and x2 modes.

Figure 541. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

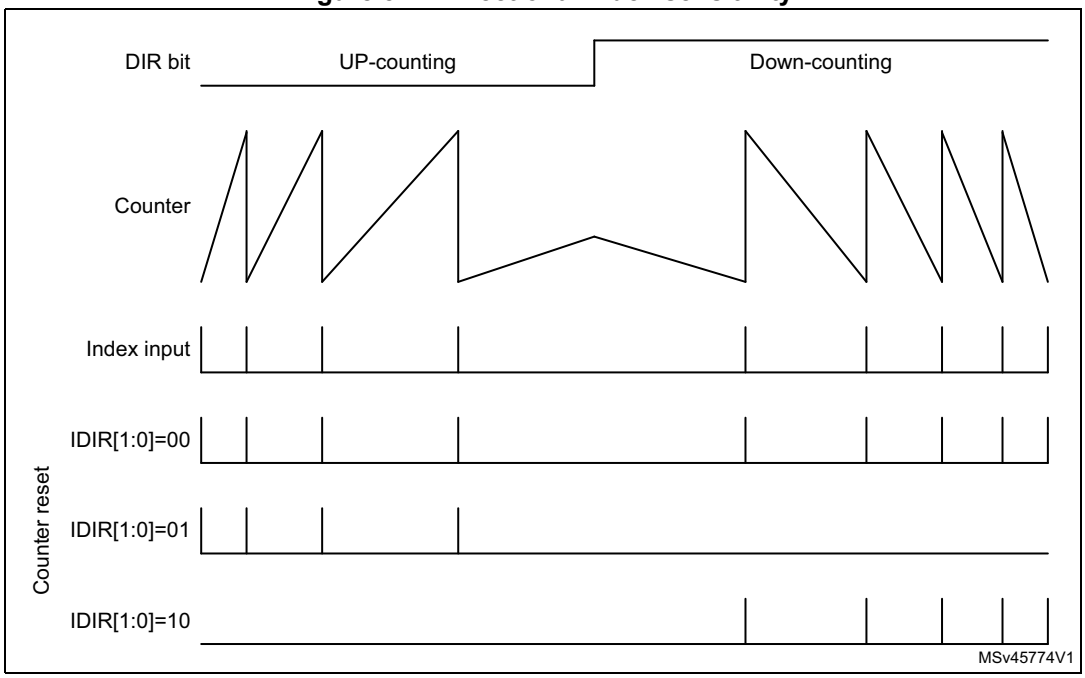


Directional index sensitivity

The IDIR[1:0] bitfield in the TIM_ECR register allows the index to be active only in a selected counting direction.

Figure 542 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

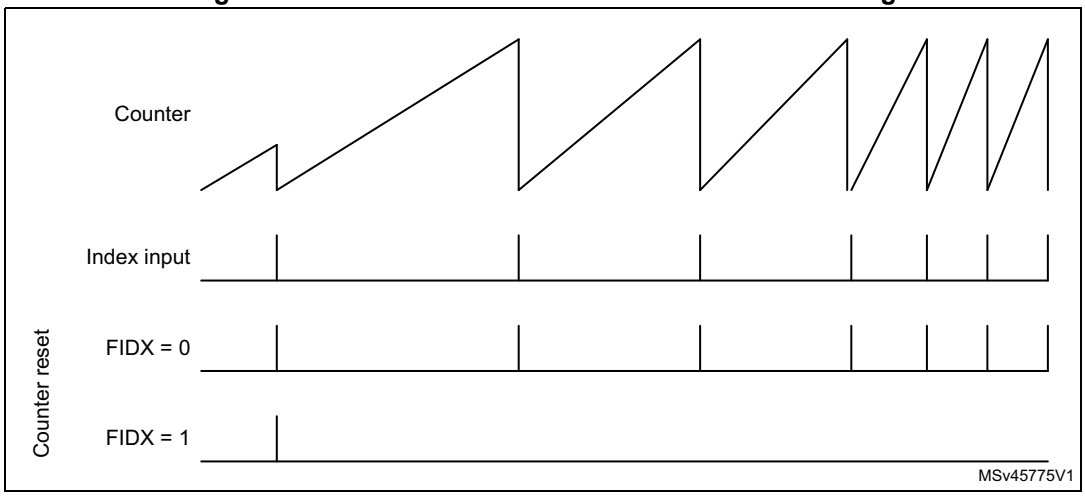
Figure 542. Directional index sensitivity



Special first index event management

The FIDX bit in the TIM_ECR register allows the index to be taken only once, as shown on *Figure 543*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 543. Counter reset as function of FIDX bit setting



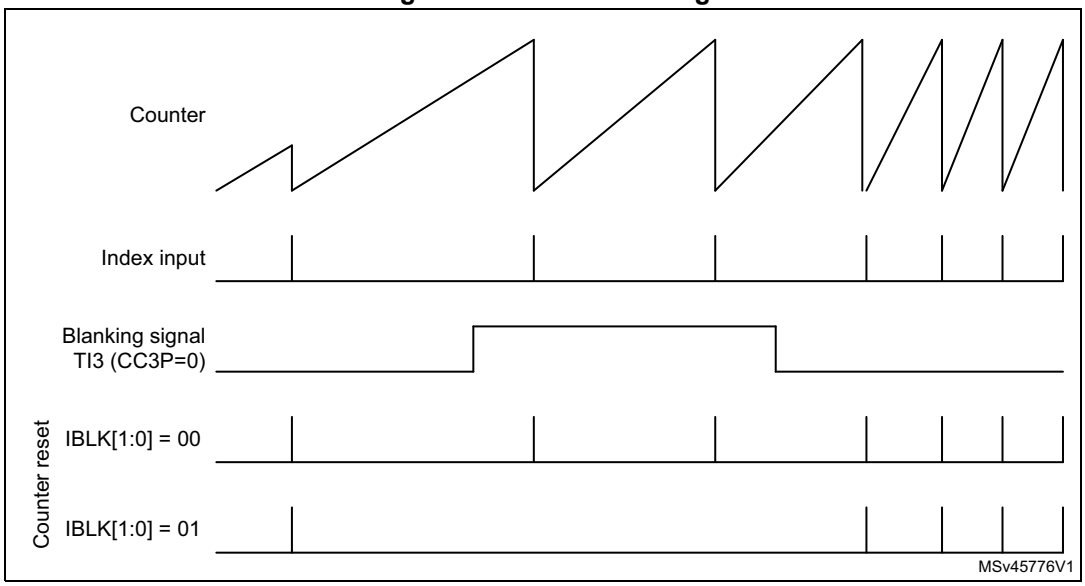
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 544* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIM_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 544. Index blanking



Index management in nonquadrature mode

Figure 545 and *Figure 546* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[4:0] bitfield is equal to 01010, 01011, 01100, 01101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 545. Index behavior in clock + direction mode, IPOS[0] = 1

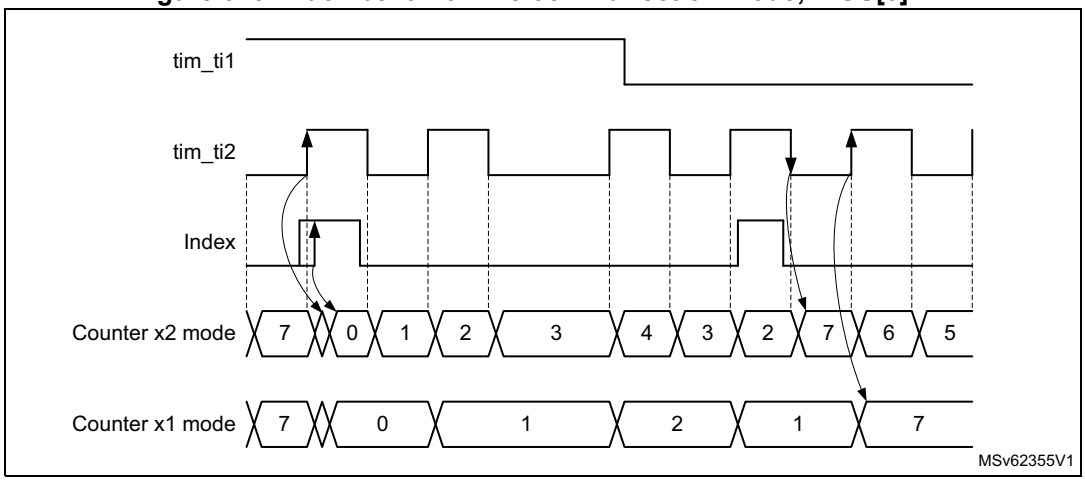
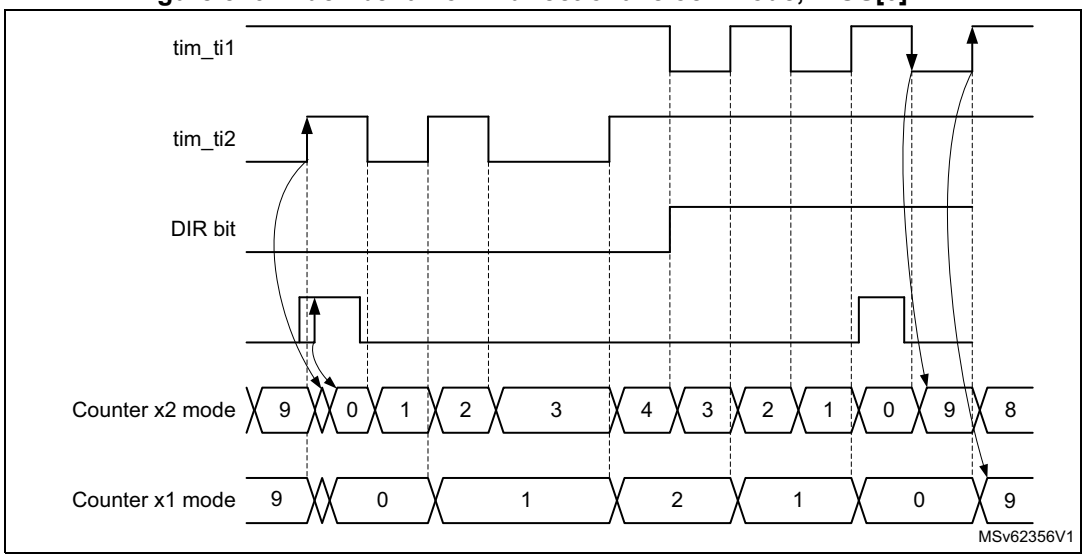


Figure 546. Index behavior in directional clock mode, IPOS[0] = 1

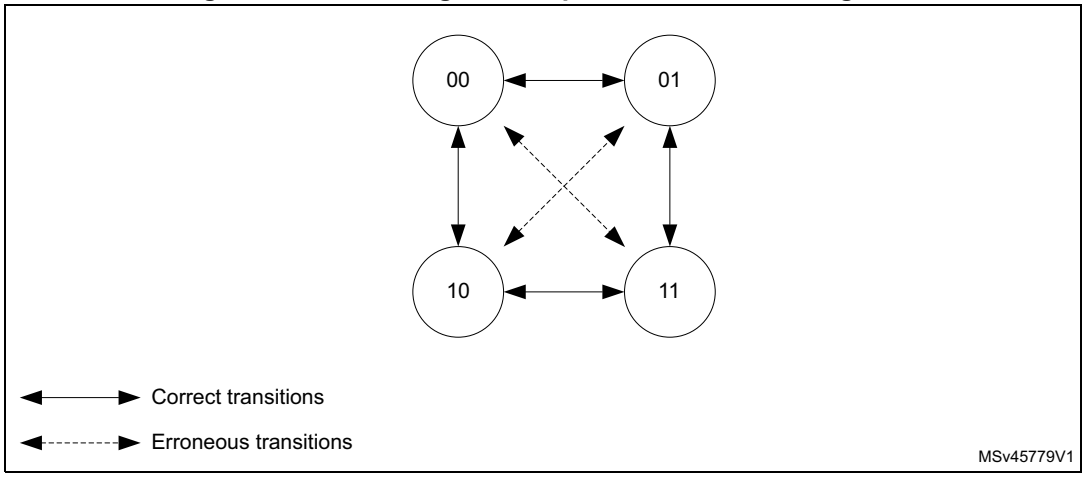


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 547.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status

register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 547. State diagram for quadrature encoded signals



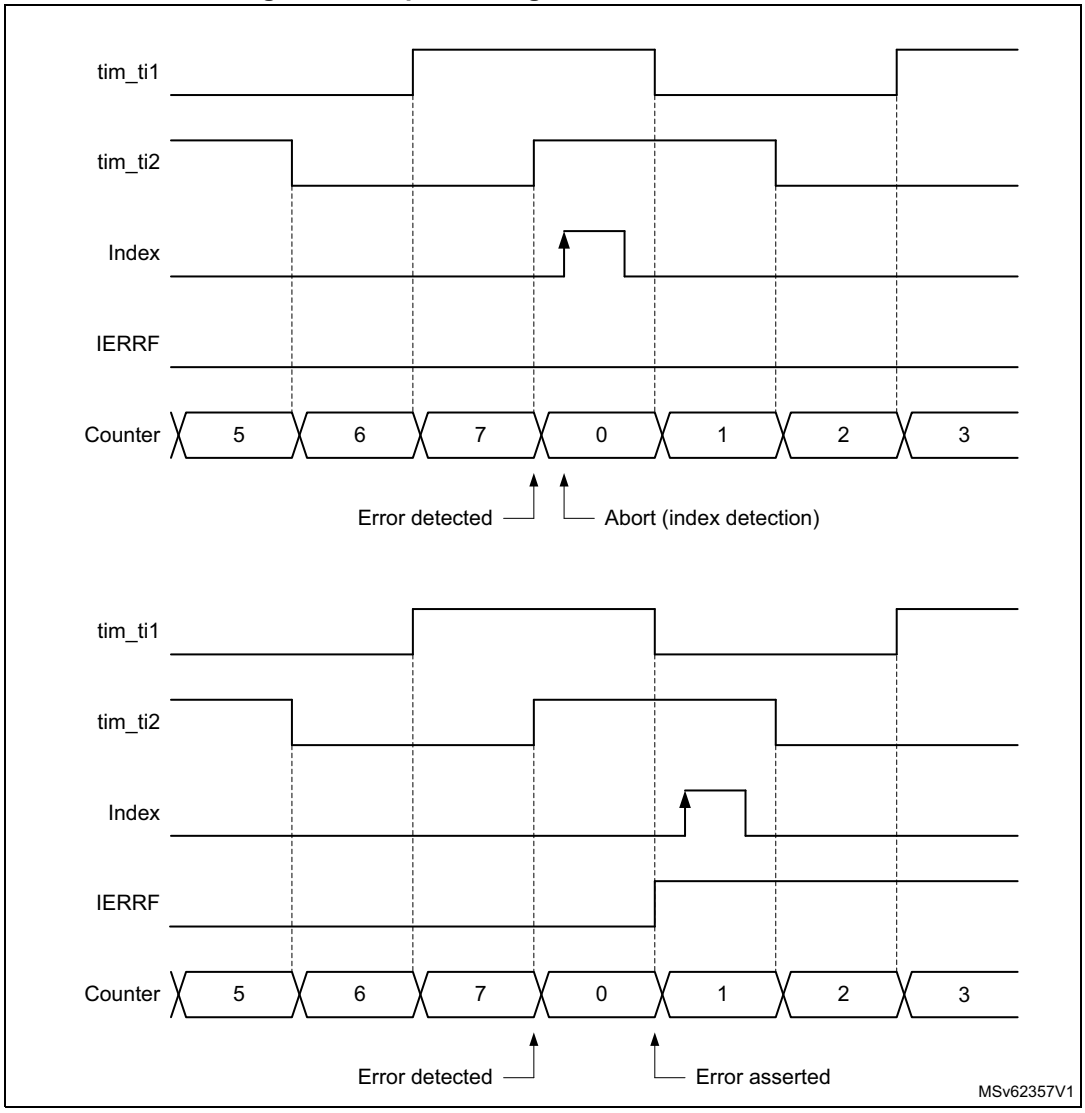
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIM_ARR to 0 or decremented from 0 to TIM_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIM_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIM_ARR = 3999 + 1 = 4000.

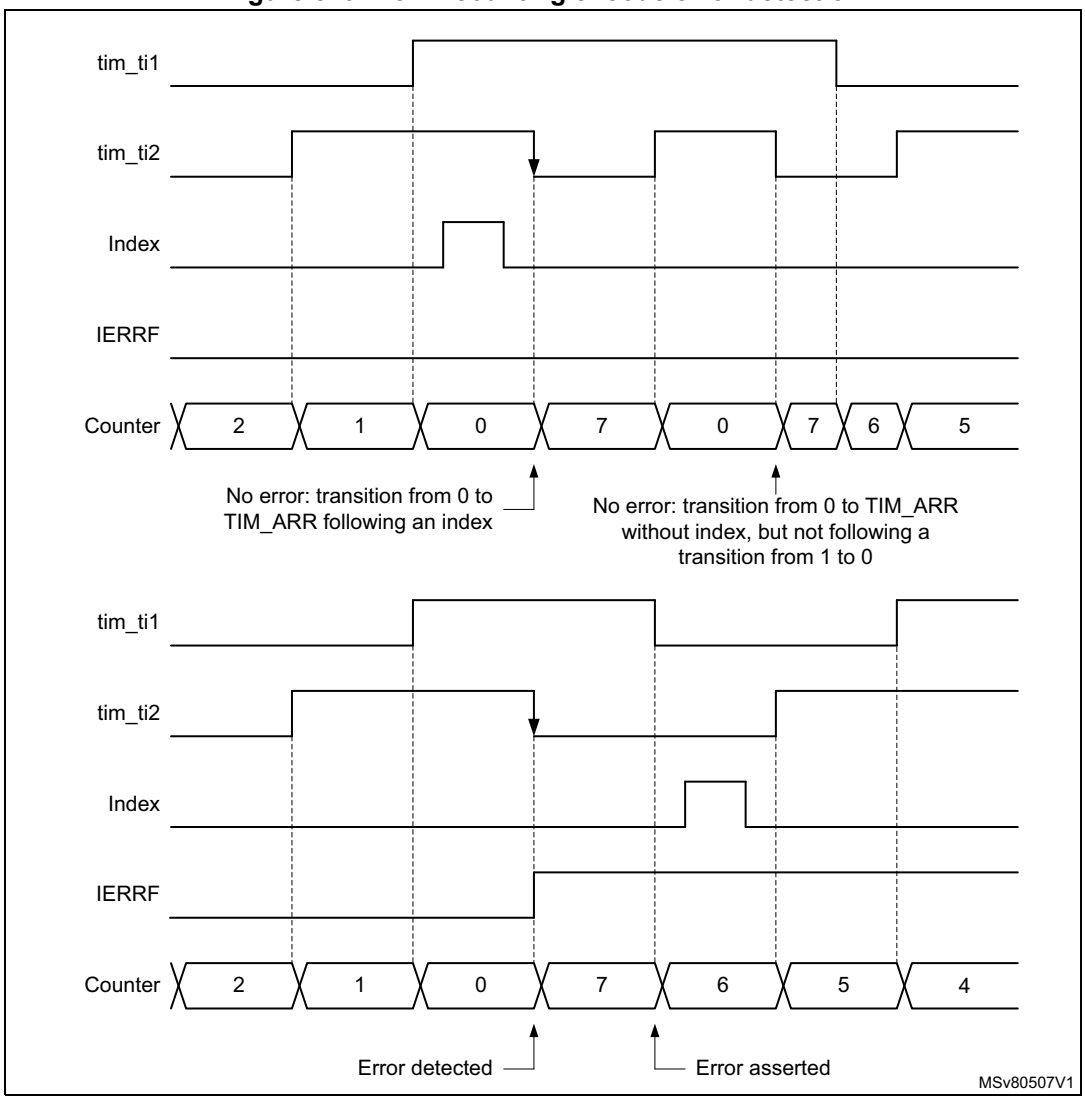
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 548*.

Figure 548. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 549*, to avoid any false error detection in case the encoder dithers between TIM_ARR and 0 immediately after the index detection.

Figure 549. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIM_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIM_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIM_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIM_DIER register.

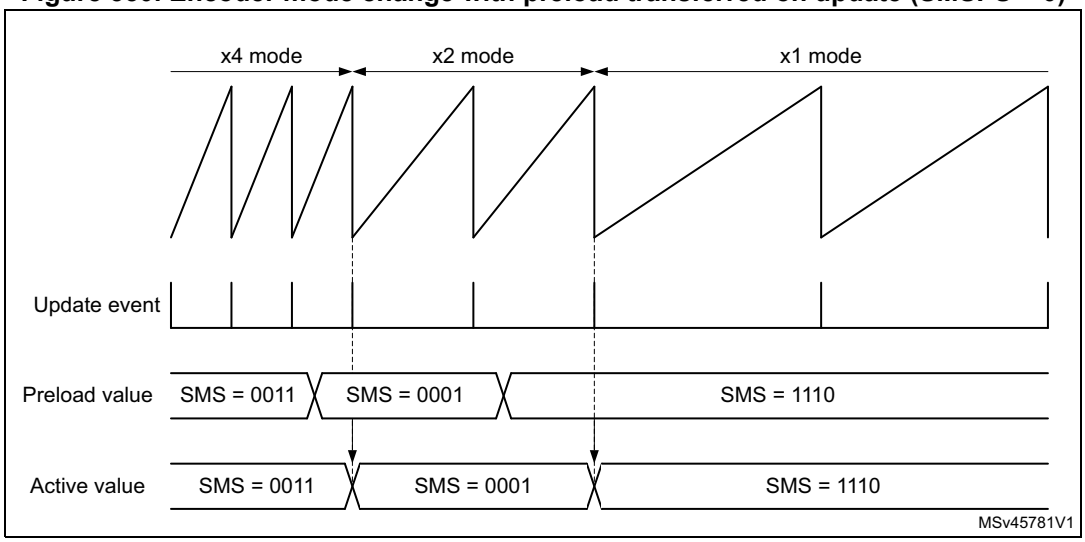
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 550*.

For this purpose, the SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value can be selected with the SMSPS bit in the TIM_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 550. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register.It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

41.3.29 Encoder with built-in debouncer

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 551* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there's no edge on input 2.

The *Figure 552* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between 2 positions.

These 2 modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 551. Encoder with built-in debouncer, x2 mode

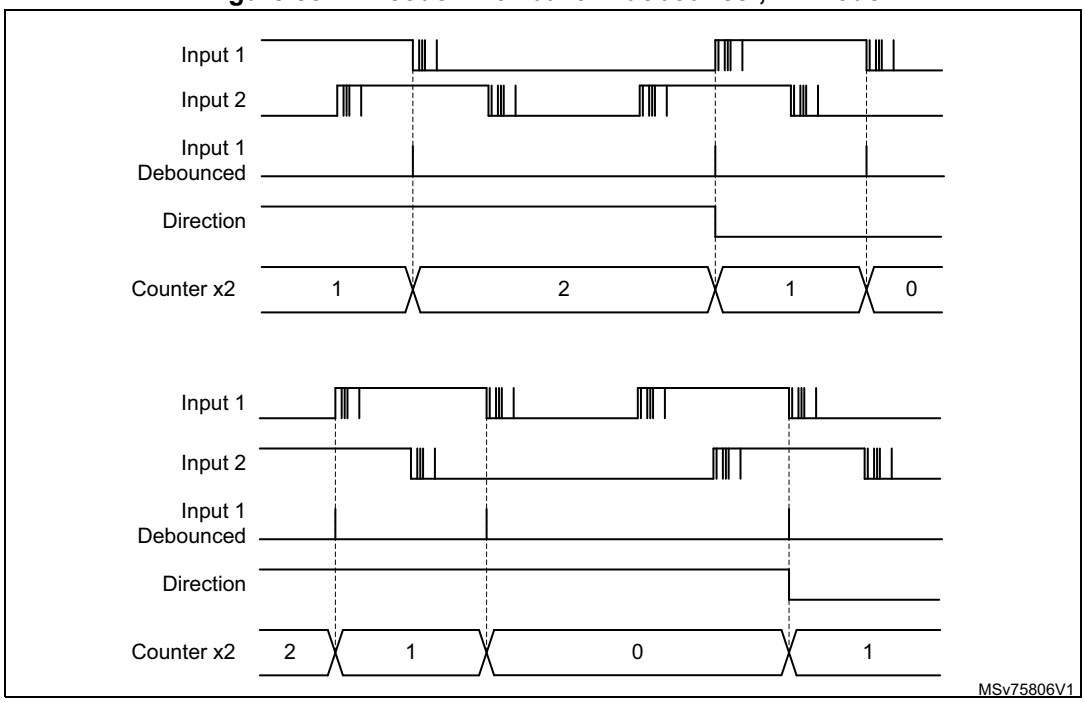
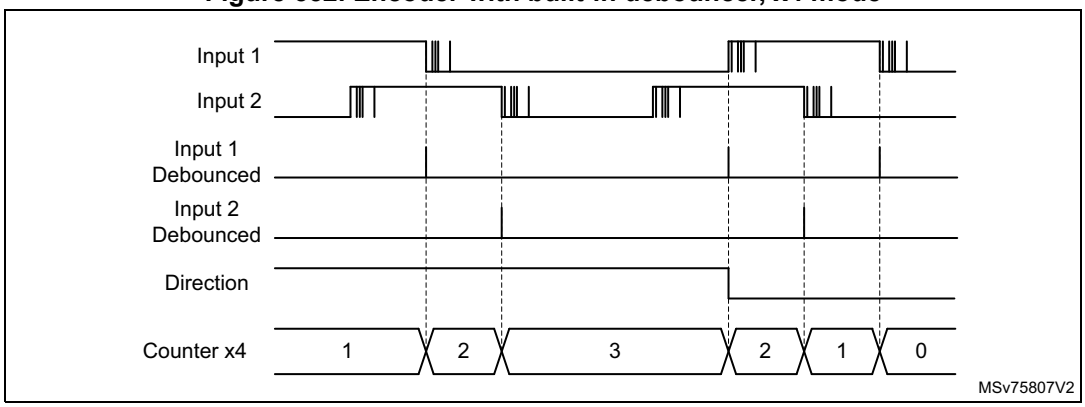


Figure 552. Encoder with built-in debouncer, x4 mode



41.3.30 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIM_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIM_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

41.3.31 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

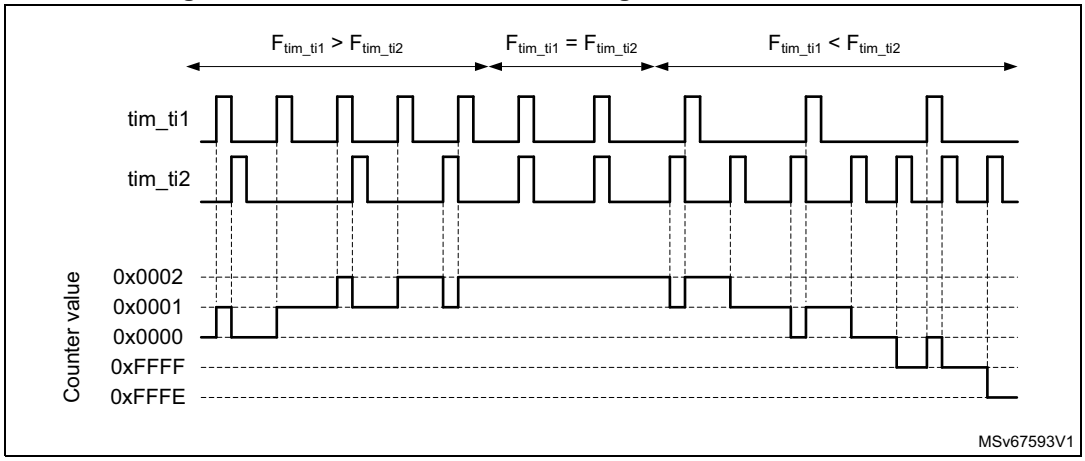
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 553*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value changes as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value dithers between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occuring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 553* below).

Figure 553. Clock drift measure using directional clock mode



41.3.32 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

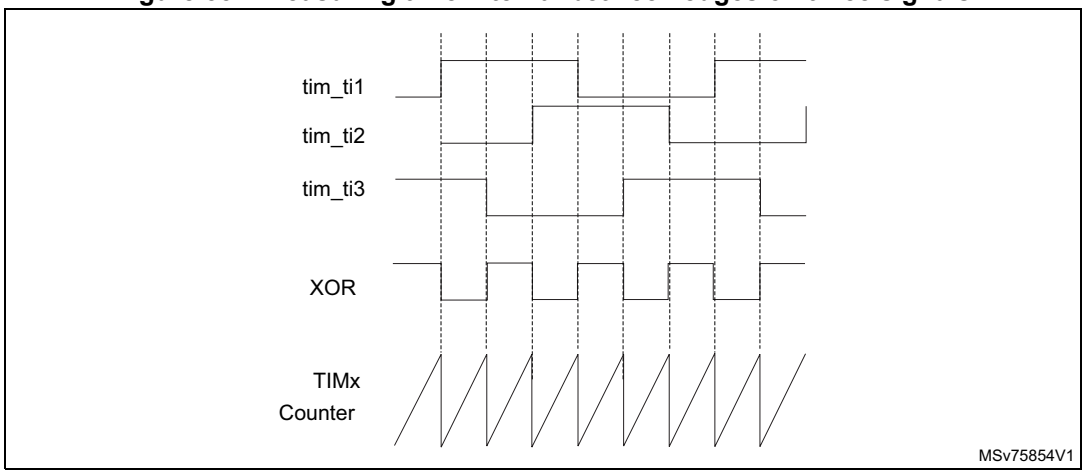
There is no latency between the UIF and UIFCPY flags assertion.

41.3.33 Timer input XOR function

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 554*.

Figure 554. Measuring time interval between edges on three signals



The XORPS control bit selects the position of the XOR gate, as shown on the *Figure 555* and *Figure 556* below, when the TI1S bit is set.

Figure 555. XOR gate connection with TI1S = 1 and XORPS = 0

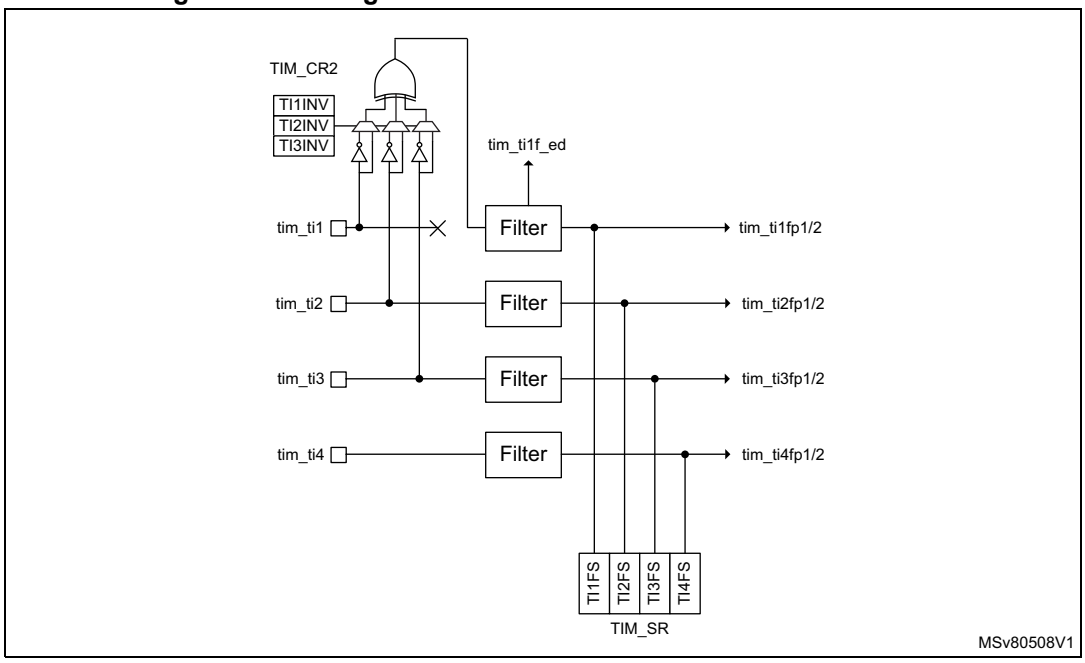
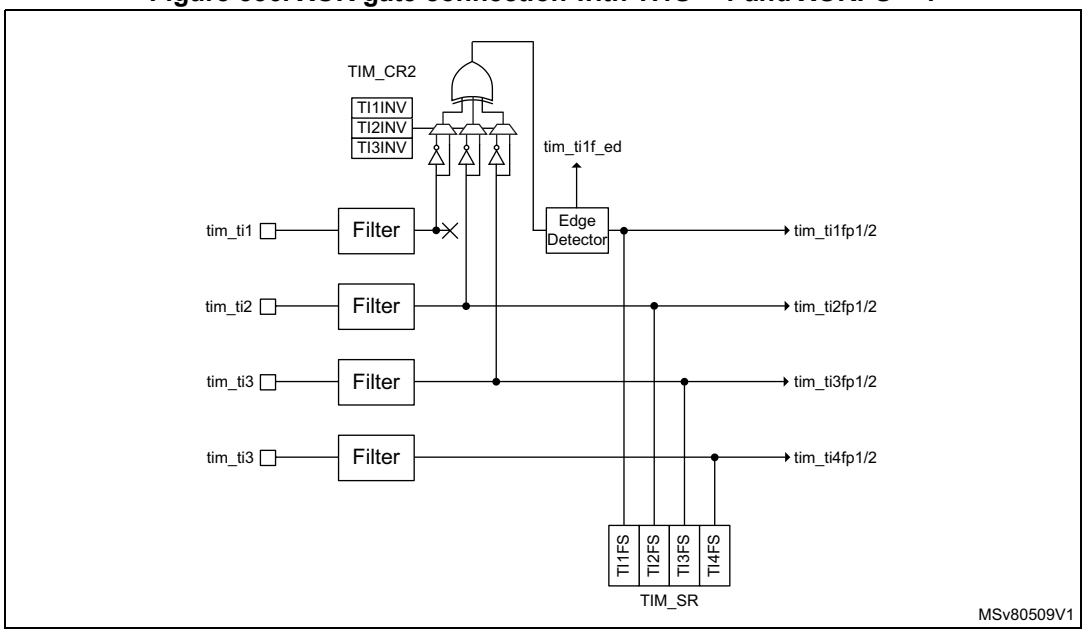


Figure 556. XOR gate connection with TI1S = 1 and XORPS = 1



Three control bits T1INV, T2INV and T3INV in the TIM_CR2 register can invert the tim_in1, tim_in2 and tim_in3 signals prior entering the XOR gate.

The status of the tim_in1, tim_in2, tim_in3 and tim_in4 signals can be software polled using the TI1FS, TI2FS, TI3FS and TI4FS bits in the TIM_SR register (tim_ti4 is not represented on figures above and do not enter into the XOR gate).

41.3.34 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as “interfacing timer” in *Figure 557*. The “interfacing timer” captures the three timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIM_CR2 register, with XORPS bit = 0).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 483*). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

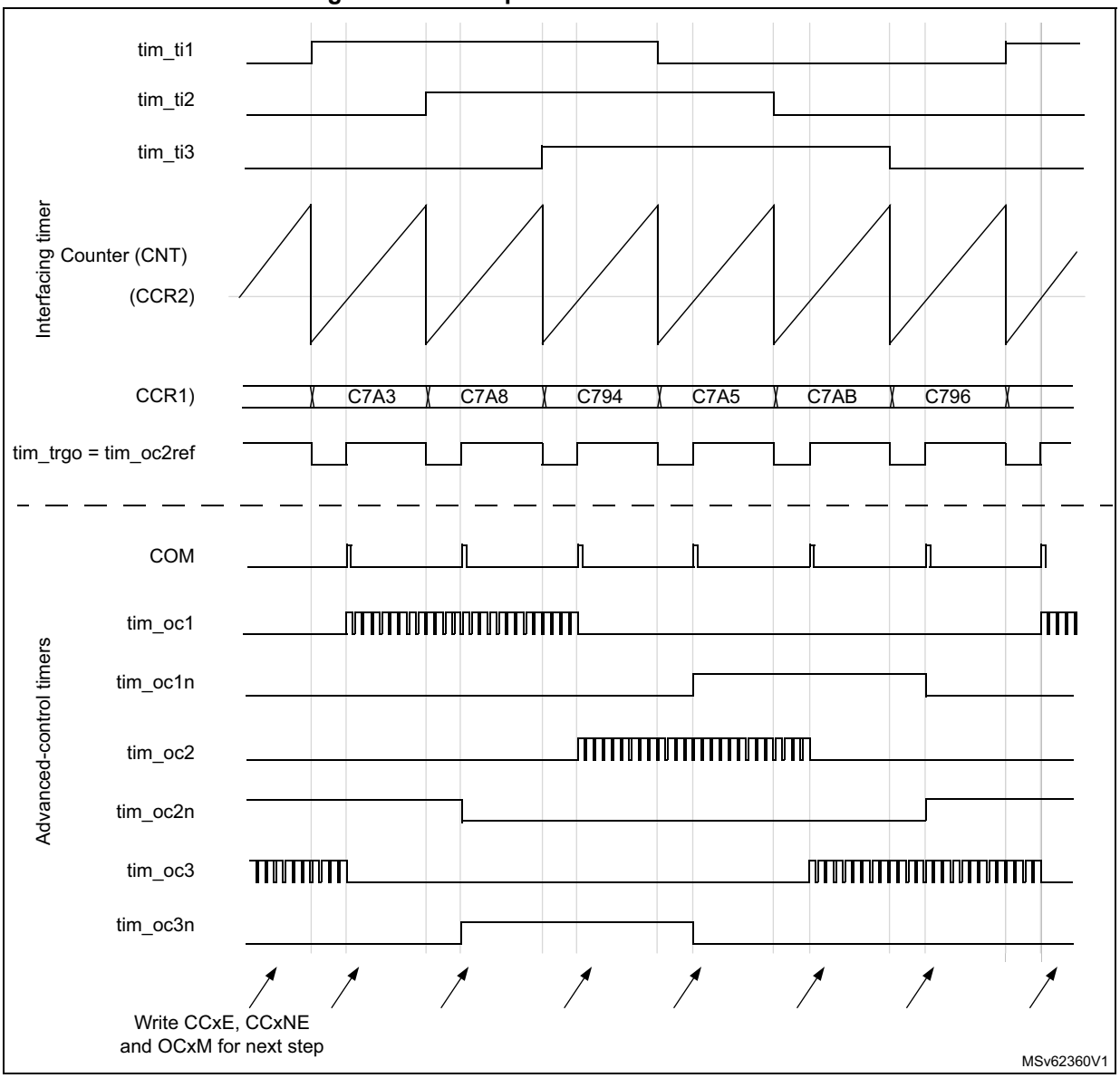
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIM_CR2 register to 1.
- Program the time base: write the TIM_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIM_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIM_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIM_CR2 register to 101.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIM_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIM_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 557 describes this example.

Figure 557. Example of Hall sensor interface



41.3.35 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 42.4.27: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

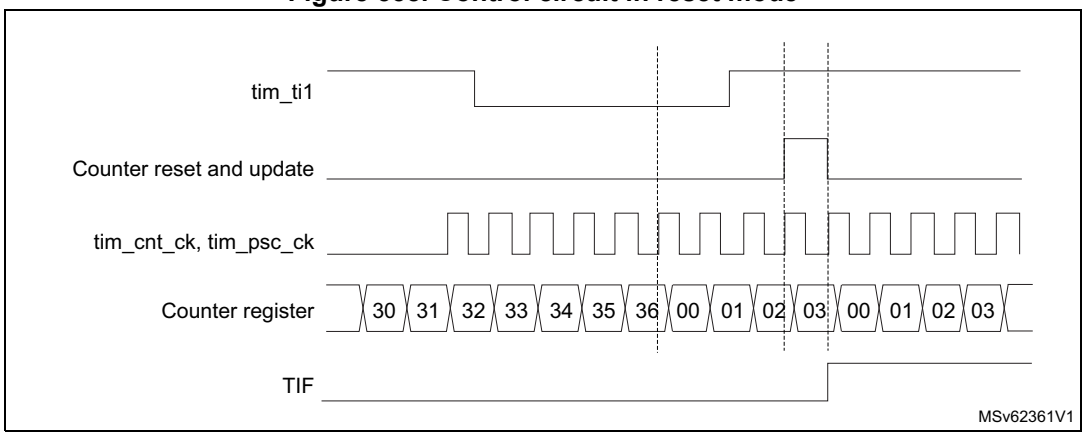
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
- Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 558. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

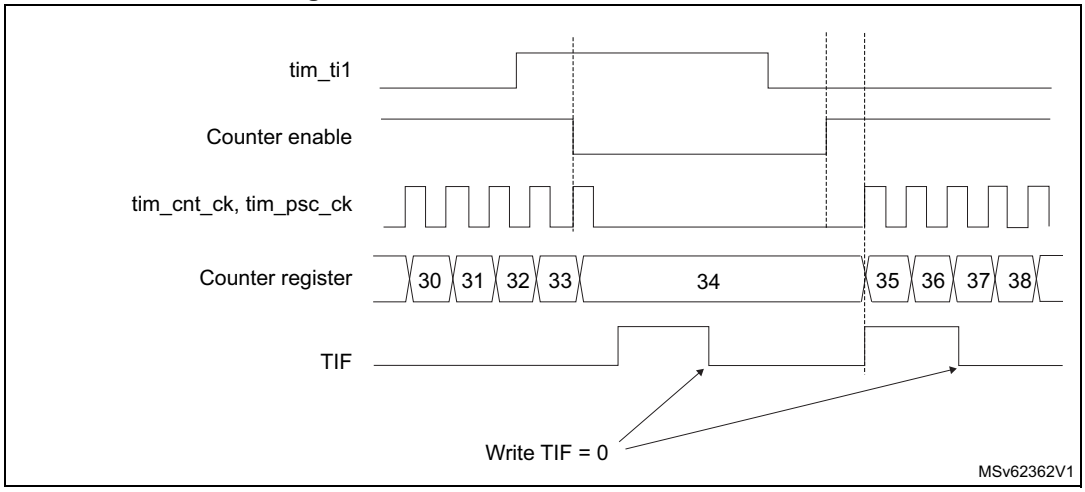
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
- Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 559. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

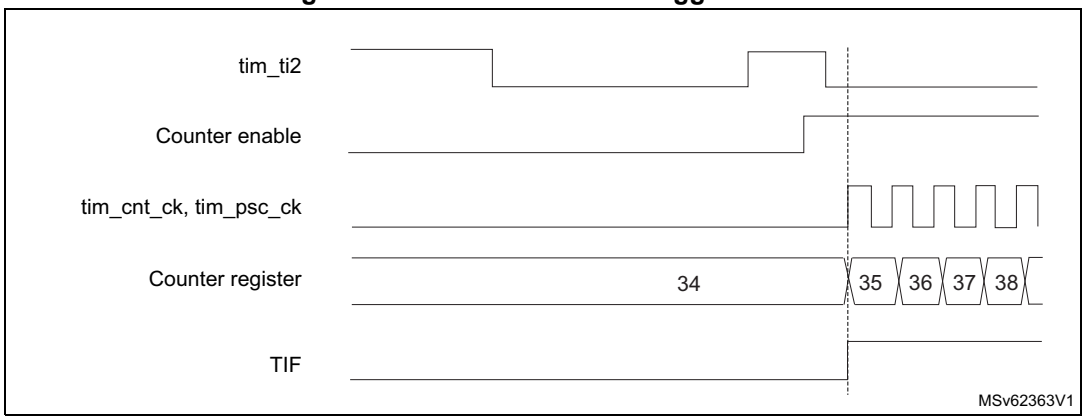
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 560. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIM_SMCR register.

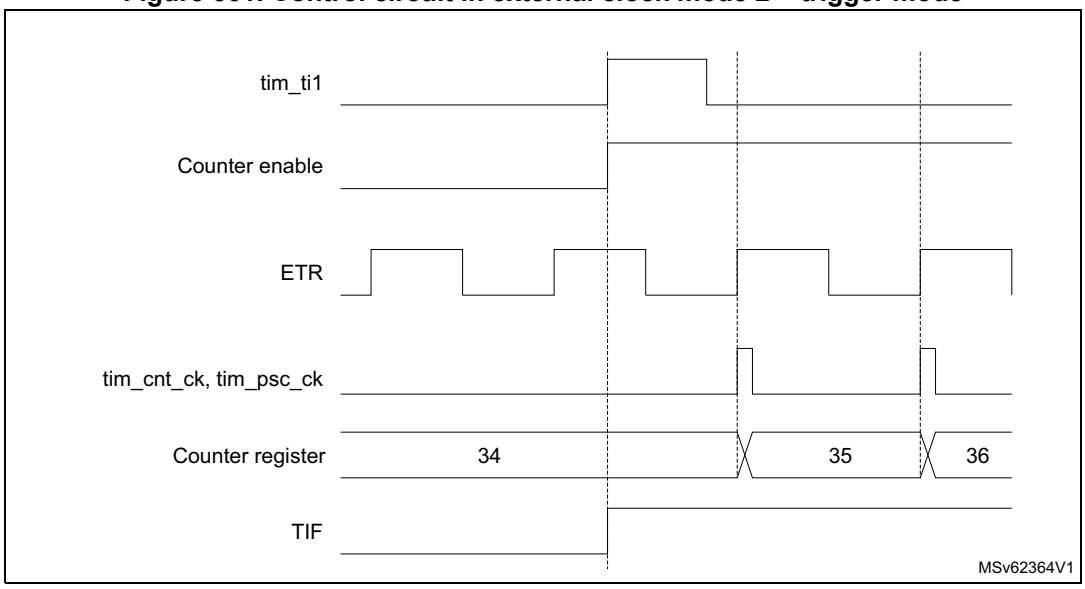
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIM_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIM_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 561. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

41.3.36 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

Refer to MMS2[4:0] bitfield description in TIM_CR2 for the complete list of triggering options.

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 24 possible events, which can be selected using the MMS2[4:0] bits in the TIM_CR2 register.

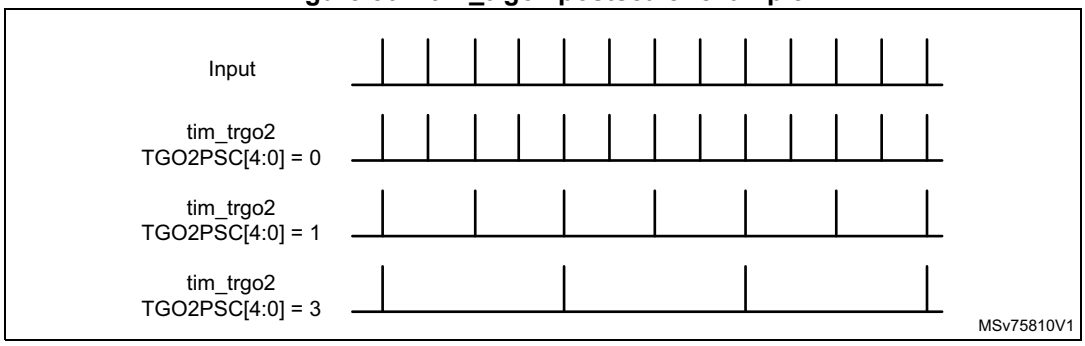
An example of an application for 3-phase motor drives is given in *Figure 501*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

It is possible to reduce the ADC triggering rate with the tim_tro2 post scaler. This is done by periodically discarding tim_trgo2 events, as per the value programmed in TGO2PSC[4:0] bitfield in TIM_CR1, as shown on *Figure 562* below.

Figure 562. tim_trgo2 postscaler example



The post scaler is able to operate with:

– pulses as shown on *Figure 562* above, when selecting for instance MMS2[4:0] = 10001 to have a trigger on tim_oc7refc rising rising or falling edges

– the tim_ocxrefc signal itself (ie with user-defined duty cycle), when selecting for instance MMS2[4:0] = 10000 to have a trigger on tim_oc7refc

41.3.37 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period:

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period:

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

41.3.38 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (in words).

The DBA[4:0] bits in the TIM_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register:

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 41.6.34: TIM DMA control register (TIM_DCR)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (see note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 2, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

41.3.39 TIM1/TIM8/TIM20 DMA requests

The TIM1/TIM8/TIM20 can generate a DMA request, as shown in the table below.

Table 470. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE

Table 470. DMA request (continued)

DMA request signal	DMA request	Enable control bit
tim_cc4_dma	Capture/compare 4	CC4DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

41.3.40 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIM_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

41.4 TIM1/TIM8/TIM20 low-power modes

Table 471. Effect of low-power modes on TIM1/TIM8/TIM20

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

41.5 TIM1/TIM8/TIM20 interrupts

The TIM1/TIM8/TIM20 can generate multiple interrupts, as shown in *Table 472*.

Table 472. Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD		Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC		Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
		Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
		Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
		Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI_COM_DIR_IDX	TIM_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	TIM_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK_TERR_IERR	TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
		Break2	B2IF		write 0 in B2IF	Yes	No
		System Break	SBIF		write 0 in SBIF	Yes	No
	TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
	TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

41.6 TIM1/TIM8/TIM20 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

41.6.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	TGO2PSC[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **TGO2PSC[4:0]**: Trigger output 2 postcaler

The postscaler discards periodically events sent on the tim_trgo2 output, to reduce the ADC triggering rate.
The number of events on tim_trgo2 is divided by (TGO2PSC[4:0] + 1).
When TGO2PSC[4:0] = 0x0, the tim_trgo2 event rate is not affected.
With TGO2PSC[4:0] = 0x1, one event out of 2 is sent on tim_trgo2, with TGO2PSC[4:0] = 0x2, one event out of 3 is sent on tim_trgo2, up to 1 event out of 31 when TGO2PSC[4:0] = 0x1F,...

Note: The postscaler is only applicable when tim_trgo2 transfers a pulse (reset, update, compare pulse). When the tim_trgo2 outputs a level-based information (compare signal or enable bit for gated mode), the postscaler must be set set to 0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Autoreload preload enable
0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.
Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

41.6.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	MMS2[4:0]					OIS7	OIS6	Res.	OIS5
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 **TI3INV**: tim_ti3 signal inversion on XOR gate input

0: tim_ti3 non-inverted

1: tim_ti3 inverted

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input

0: tim_ti2 non-inverted

1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input

0: tim_ti1 non-inverted

1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 41.3.37* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection

This bit defines the position of the XOR gate with respect to input filters.

0: XOR gate placed before TI1 filter

1: XOR gate placed after TI1, TI2 and TI3 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:20 **MMS2[4:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

00000:**Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

00001:**Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIM_SMCR register).

00010:**Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

00011:**Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

00100:**Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

00101:**Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

00110:**Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

00111:**Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

01000:**Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

01001:**Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

01010:**Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

01011:**Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

01100:**Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

01101:**Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

01110:**Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

01111:**Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

10000: **Compare** - tim_oc7refc signal is used as trigger output (tim_trgo2)

10001: **Compare Pulse** - tim_oc7refc rising or falling edges generate pulses on tim_trgo2

10010: **Compare Pulse** tim_oc4refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10011: **Compare Pulse** - tim_oc4refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

10100: **Compare Pulse** -tim_oc5refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10101: **Compare Pulse** - tim_oc5refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

10110: **Compare Pulse** - tim_oc6refc or tim_oc7efc rising edges generate pulses on tim_trgo2

10111: **Compare Pulse** - tim_oc6refc rising or tim_oc7refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 **OIS7**: Output idle state 7 (tim_oc7 output)

Refer to OIS1 bit

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 (after a dead-time) when MOE = 0
1: tim_oc1 = 1 (after a dead-time) when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
This bit enables the insertion of a XOR gate to combine tim_ti1_in[15:0], tim_ti2_in[15:0], and tim_ti3_in[15:0] inputs into tim_ti1.
The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE, CCxP, CCxNP and OCxM bits are not preloaded

1: CCxE, CCxNE, CCxP, CCxNP and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

41.6.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	SMS[4:3]	
		rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:26 **SETPS[3:0]**: Synchronous external trigger prescaler

The linear synchronous prescaler divides the frequency of the tim_etrf signal, by a factor (SETPS[3:0] + 1).

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[4:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

Bits 19:18 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Bits 17:16, 2:0 **SMS[4:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000:Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001:Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010:Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011:Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100:Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101:Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110:Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111:External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000:Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001:Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010:Encoder mode: Clock plus direction, x2 mode.

01011:Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

01100:Encoder mode: Directional Clock, x2 mode.

01101:Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110:Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111:Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000:Quadrature encoder with built-in debouncer: x2 mode.

10001:Quadrature encoder with built-in debouncer: x4 mode.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

41.6.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

41.6.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	ODS	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	CC7IF	CC6IF	CC5IF
r	r	r	r			r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0		rc_w0	rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B2GF	BGF	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 31 **TI4FS**: tim_ti4f status

This flag indicates the value of the tim_ti4f signal (after the digital filtering stage), for polling purpose.
0: tim_ti4f signal is low
1: tim_ti4f signal is high

Bit 30 **TI3FS**: tim_ti3f status

This flag indicates the value of the tim_ti3f signal (after the digital filtering stage), for polling purpose.
0: tim_ti3f signal is low
1: tim_ti3f signal is high

Bit 29 **TI2FS**: tim_ti2f status

This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status

This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **ODS**: Output disable status

This bit indicates the global output state when a break or break2 event occured, to discriminate the source.
0: Idle state: break was triggered (or MOE was written to 0), and OIS states are forced on tim_ocx and tim_ocxn outputs
1: Disabled state: break2 was triggered, tim_ocx and tim_ocxn outputs are inactive

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag

This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag

This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag

This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CC7IF**: Compare 7 interrupt flag
Refer to CC1IF description
Note: Channel 7 can only be configured as output.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

Bit 15 **B2GF**: Break 2 generation flag
This flag is set by hardware as soon as a break 2 event is forced by software, using the B2G bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break 2 event occurred.
1: A break 2 event was forced with the B2G bit

Bit 14 **BGF**: Break generation flag
This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break event occurred.
1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.
–When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
–When CNT is reinitialized by a trigger event (refer to *Section 41.6.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

41.6.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIM_ARR) if DIR = 1 (downcounting).

41.6.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter
Refer to IC1F[3:0] bit description

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler
Refer to IC1PSC[1:0] bit description

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

41.6.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output compare 2 clear enable

0: tim_oc2ref is not affected by the tim_ocref_clr_int signal

1: tim_oc2ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 41.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode

Refer to OC1M[4:0] bit description

Bit 11 **OC2PE**: Output compare 2 preload enable

0: Preload register on TIM_CCR2 disabled. TIM_CCR2 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR2 enabled. Read/Write operations access the preload register. TIM_CCR2 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC2FE**: Output compare 2 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC2 behaves normally depending on counter and CCR2 values even when the trigger is ON. The minimum delay to activate CC2 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC2 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC2 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 41.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000:Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs.This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001:Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010:Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011:Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100:Force inactive level - tim_oc1ref is forced low.

00101:Force active level - tim_oc1ref is forced high.

00110:PWM mode 1 - In upcounting, channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref = 0) as long as TIM_CNT>TIM_CCR1 else active (tim_oc1ref = 1).

00111:PWM mode 2 - In upcounting, channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active. In downcounting, channel 1 is active as long as TIM_CNT>TIM_CCR1 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 465* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 464* for details.

10000-10001:Combined PWM mode 3 and 4. Refer to *Table 465* for details.

10010-11001: Asymmetric PWM mode 3 to 10. Refer to *Table 464* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

41.6.9 TIM capture/compare mode register 2 (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter
Refer to IC3F[3:0] bit description

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to IC3PSC[1:0] bit description

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter
This bitfield defines the frequency used to sample tim_ti3 input and the length of the digital filter applied to tim_ti3. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
0000: No filter, sampling is done at f_{DTS2}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS2}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS2}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS2}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS2}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS2}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS2}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS2}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS2}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS2}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS2}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS2}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS2}/32$, N = 8

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler
This bitfield defines the ratio of the prescaler acting on CC3 input (tim_ic3). The prescaler is reset as soon as CC3E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

41.6.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

0: tim_oc4ref is not affected by the tim_ocref_clr_int signal

1: tim_oc4ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 41.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC4M[4:0]**: Output compare 4 mode

Refer to OC3M[4:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

0: Preload register on TIM_CCR4 disabled. TIM_CCR4 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR4 enabled. Read/Write operations access the preload register. TIM_CCR4 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC4FE**: Output compare 4 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC4 behaves normally depending on counter and CCR4 values even when the trigger is ON. The minimum delay to activate CC4 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC4 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC4 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

0: tim_oc3ref is not affected by the tim_ocref_clr_int signal

1: tim_oc3ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 41.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC3M[4:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

00000:Frozen - The comparison between the output compare register TIM_CCR3 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00010:Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00011:Toggle - tim_oc3ref toggles when TIM_CNT = TIM_CCR3.

00100:Force inactive level - tim_oc3ref is forced low.

00101:Force active level - tim_oc3ref is forced high.

00110:PWM mode 1 - In upcounting, channel 3 is active as long as TIM_CNT<TIM_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIM_CNT>TIM_CCR3 else active (tim_oc3ref = 1).

00111:PWM mode 2 - In upcounting, channel 3 is inactive as long as TIM_CNT<TIM_CCR3 else active. In downcounting, channel 3 is active as long as TIM_CNT>TIM_CCR3 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010:Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIM_ECR.

01011:Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 465* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 464* for details.

10000-10001:Combined PWM mode 3 and 4. Refer to *Table 465* for details.

10010-11001: Asymmetric PWM mode 3 to 10. Refer to *Table 464* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

0: Preload register on TIM_CCR3 disabled. TIM_CCR3 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR3 enabled. Read/Write operations access the preload register. TIM_CCR3 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC3FE**: Output compare 3 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC3 behaves normally depending on counter and CCR3 values even when the trigger is ON. The minimum delay to activate CC3 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC3 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC3 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

41.6.11 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC7P	CC7E	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
						rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CC7P**: Capture/compare 7 output polarity
Refer to CC1P description

Bit 24 **CC7E**: Capture/compare 7 output enable
Refer to CC1E description

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.
Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Note: In input mode, this bit is preloaded as per CCPC bit in TIM_CR2.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:the configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Note: In input mode, this bit is preloaded as per CCPC bit in TIM_CR2.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 473* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 473. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>tim_ocx output state</th><th>tim_ocxn output state</th></tr>
<tr><td rowspan="8">1</td><td rowspan="8">X</td><td>0</td><td>0</td><td>0</td><td colspan="2">Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0, tim_ocxn = 0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td colspan="2">Off-State (output enabled with inactive state)
tim_ocx=CCxP, tim_ocxn=CCxNP</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0</td><td>tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_ocxn = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_ocx = CCxP</td><td>tim_ocxref + Polarity
tim_ocxn = tim_ocxref x or CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Off-State (output enabled with inactive state)
tim_ocxn = CCxNP</td></tr>
<tr><td rowspan="6">0</td><td>0</td><td rowspan="6">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z).</td></tr>
<tr><td rowspan="5">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered).
Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration).
Note: tim_brk2 can only be used if OSSI = OSSR = 1.</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

41.6.12 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in the TIM_CR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

41.6.13 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

41.6.14 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 41.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

41.6.15 TIM repetition counter register (TIM_RCR)

Address offset: 0x030

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

41.6.16 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

41.6.17 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

41.6.18 TIM capture/compare register 3 (TIM_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc3 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.
If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIM_CCR3 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

41.6.19 TIM capture/compare register 4 (TIM_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIM_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

41.6.20 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional
Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BK2E bit is set.
Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection, see *Figure 507: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled
Note: The BRKIN2 must only be used with OSSR = OSSI = 1.
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).
Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BK2E bit is set.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM_CCER register).
See OC/OCN enable description for more details (*Section 41.6.11: TIM capture/compare enable register (TIM_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to tim_sys_brk and BKIN sources, as per *Figure 507: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Any write operation to this bit takes a delay of 1 tim_ker_ck clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 41.6.11: TIM capture/compare enable register (TIM_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 41.6.11: TIM capture/compare enable register (TIM_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BK2BID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.
DTG[7:5] = 110 => DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.
DTG[7:5] = 111 => DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

41.6.21 TIM capture/compare register 5 (TIM_CCR5)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	GC5C4	GC5C4O	GC5C3O	GC5C2O	GC5C1O	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3

Distortion on channel 3 output:

0: No effect of tim_oc5ref on tim_oc3refc

1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2

Distortion on channel 2 output:

0: No effect of tim_oc5ref on tim_oc2refc

1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1

Distortion on channel 1 output:

0: No effect of oc5ref on oc1refc

1: oc1refc is the logical AND of oc1ref and oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 28 **GC5C4**: Group channel 5 and channel 4

Distortion on channel 1 output:

0: No effect of oc5ref on oc4refc

1: oc4refc is the logical AND of oc4ref and oc5ref

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 27 **GC5C4O**: Group channel 5 and channel 4 (ORed)

Distortion on channel 4 output:

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

0: No effect of oc5ref on oc4refc

1: oc4refc is the logical OR of oc4ref and oc5ref

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 26 **GC5C3O**: Group channel 5 and channel 3 (ORed)

Distortion on channel 3 output:

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).

0: No effect of oc5ref on oc3refc

1: oc4refc is the logical OR of oc3ref and oc5ref

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 25 **GC5C2O**: Group channel 5 and channel 2 (ORed)
Distortion on channel 2 output:
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).
0: No effect of oc5ref on oc2refc
1: oc4refc is the logical OR of oc2ref and oc5ref
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 24 **GC5C1O**: Group channel 5 and channel 1 (ORed)
Distortion on channel 1 output:
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload is enabled with OC5PE in TIM_CCMR3).
0: No effect of oc5ref on oc1refc
1: oc4refc is the logical OR of oc1ref and oc5ref
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: compare 5 value
CCR5 is the value to be loaded in the actual compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC5PE). Else the preload value is copied in the active compare 5 register when an update event occurs.
The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

41.6.22 TIM capture/compare register 6 (TIM_CCR6)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: Compare 6 value

CCR6 is the value to be loaded in the actual compare 6 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC6PE). Else the preload value is copied in the active compare 6 register when an update event occurs.

The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc6 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

41.6.23 TIM capture/compare mode register 3 (TIM_CCMR3)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC6M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC5M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6CE	OC6M[2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

0: tim_oc6ref is not affected by the tim_ocref_clr_int signal

1: tim_oc6ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 41.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC6M[4:0]**: Output compare 6 mode

Refer to OC5M[4:0] bit description

Bit 11 **OC6PE**: Output compare 6 preload enable

0: Preload register on TIM_CCR6 disabled. TIM_CCR6 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR6 enabled. Read/Write operations access the preload register. TIM_CCR6 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 10 **OC6FE**: Output compare 6 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC6 behaves normally depending on counter and CCR6 values even when the trigger is ON. The minimum delay to activate CC6 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC6 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC6 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

0: tim_oc5ref is not affected by the tim_ocref_clr_int signal

1: tim_oc5ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 41.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC5M[4:0]**: Output compare 5 mode

These bits define the behavior of the output reference signal tim_oc5ref5. tim_oc5ref is active high whereas tim_oc5 active level depends on CC5P bit.

00000:Frozen - The comparison between the output compare register TIM_CCR5 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 5 to active level on match. tim_oc5ref signal is forced high when the counter TIM_CNT matches the capture/compare register 5 (TIM_CCR5).

00010:Set channel 5 to inactive level on match. tim_oc5ref signal is forced low when the counter TIM_CNT matches the capture/compare register 5 (TIM_CCR5).

00011:Toggle - tim_oc5ref toggles when TIM_CNT = TIM_CCR5.

00100:Force inactive level - tim_oc5ref is forced low.

00101:Force active level - tim_oc5ref is forced high.

00110:PWM mode 1 - In upcounting, channel 5 is active as long as TIM_CNT<TIM_CCR5 else inactive. In downcounting, channel 5 is inactive (tim_oc5ref = 0) as long as TIM_CNT>TIM_CCR5 else active (tim_oc5ref = 1).

00111:PWM mode 2 - In upcounting, channel 5 is inactive as long as TIM_CNT<TIM_CCR5 else active. In downcounting, channel 5 is active as long as TIM_CNT>TIM_CCR5 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010-01011:Reserved.

01100-01101:Combined PWM mode 1 and 2. Refer to *Table 465* for details.

01110-01111:Asymmetric PWM mode 1 and 2. Refer to *Table 464* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC5PE**: Output compare 5 preload enable

0: Preload register on TIM_CCR5 disabled. TIM_CCR5 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR5 enabled. Read/Write operations access the preload register. TIM_CCR5 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC5FE**: Output compare 5 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC5 behaves normally depending on counter and CCR5 values even when the trigger is ON. The minimum delay to activate CC5 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC5 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC5 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 Reserved, must be kept at reset value.

41.6.24 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx => DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.

DTGF[7:5] = 10x => DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.

DTGF[7:5] = 110 => DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.

DTGF[7:5] = 111 => DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

41.6.25 TIM timer encoder control register (TIM_ECR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[4:0] = 00001, 00010, 00011, 01110, 01111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[4:0] = 01010, 01011, 01100, 01101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 or tim_ti4 input

00: Index always active
01: Index disabled when tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

41.6.26 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIM_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIM_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIM_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIM_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for interconnects list.

41.6.27 TIM alternate function option register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCM P20E	BKCM P19E	BKCM P18E	BKCM P17E	BKCM P16E	BKCM P15E	BKCM P14E	BKCM P13E	BKCM P12E	BKCM P11E	BKCM P10E	BKCM P9E	Res.	ETRSEL[4:2]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BKCM P4P	BKCM P3P	BKCM P2P	BKCM P1P	BKINP	BKCM P8E	BKCM P7E	BKCM P6E	BKCM P5E	BKCM P4E	BKCM P3E	BKCM P2E	BKCM P1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable

This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp20 input disabled

1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable
This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp13 input disabled
1: tim_brk_cmp13 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable
This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp12 input disabled
1: tim_brk_cmp12 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable
This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp11 input disabled
1: tim_brk_cmp11 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable
This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp10 input disabled
1: tim_brk_cmp10 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable
This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp9 input disabled
1: tim_brk_cmp9 input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 19 Reserved, must be kept at reset value.

Bits 18:14 **ETRSEL[4:0]**: etr_in source selection
These bits select the etr_in input source.
00000:tim_etr0: TIM_ETR input
00001:tim_etr1
...
11111:tim_etr31
Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific implementation.
Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIM_BKIN input polarity

This bit selects the TIM_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIM_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIM_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIM_BKIN input enable

This bit enables the TIM_BKIN alternate function input for the timer's tim_brk input. TIM_BKIN input is 'ORed' with the other tim_brk sources.

0: TIM_BKIN input disabled

1: TIM_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Refer to Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals for product specific implementation.

41.6.28 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BK2CMP20E	BK2CMP19E	BK2CMP18E	BK2CMP17E	BK2CMP16E	BK2CMP15E	BK2CMP14E	BK2CMP13E	BK2CMP12E	BK2CMP11E	BK2CMP10E	BK2CMP9E	OCRSEL[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BK2CMP20E**: tim_brk2_cmp20 enable

This bit enables the tim_brk2_cmp20 for the timer's tim_brk2 input. tim_brk2_cmp20 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp20 input disabled

1: tim_brk2_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BK2CMP19E**: tim_brk2_cmp19 enable

This bit enables the tim_brk2_cmp19 for the timer's tim_brk2 input. tim_brk2_cmp19 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp19 input disabled

1: tim_brk2_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BK2CMP18E**: tim_brk2_cmp18 enable

This bit enables the tim_brk2_cmp18 for the timer's tim_brk2 input. tim_brk2_cmp18 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp18 input disabled

1: tim_brk2_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BK2CMP17E**: tim_brk2_cmp17 enable

This bit enables the tim_brk2_cmp17 for the timer's tim_brk2 input. tim_brk2_cmp17 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp17 input disabled

1: tim_brk2_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BK2CMP16E**: tim_brk2_cmp16 enable

This bit enables the tim_brk2_cmp16 for the timer's tim_brk2 input. tim_brk2_cmp16 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp16 input disabled

1: tim_brk2_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BK2CMP15E**: tim_brk2_cmp15 enable

This bit enables the tim_brk2_cmp15 for the timer's tim_brk2 input. tim_brk2_cmp15 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp15 input disabled

1: tim_brk2_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BK2CMP14E**: tim_brk2_cmp14 enable

This bit enables the tim_brk2_cmp14 for the timer's tim_brk2 input. tim_brk2_cmp14 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp14 input disabled

1: tim_brk2_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BK2CMP13E**: tim_brk2_cmp13 enable

This bit enables the tim_brk2_cmp13 for the timer's tim_brk2 input. tim_brk2_cmp13 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp13 input disabled

1: tim_brk2_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BK2CMP12E**: tim_brk2_cmp12 enable

This bit enables the tim_brk2_cmp12 for the timer's tim_brk2 input. tim_brk2_cmp12 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp12 input disabled

1: tim_brk2_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BK2CMP11E**: tim_brk2_cmp11 enable

This bit enables the tim_brk2_cmp11 for the timer's tim_brk2 input. tim_brk2_cmp11 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp11 input disabled

1: tim_brk2_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BK2CMP10E**: tim_brk2_cmp10 enable

This bit enables the tim_brk2_cmp10 for the timer's tim_brk2 input. tim_brk2_cmp10 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp10 input disabled

1: tim_brk2_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BK2CMP9E**: tim_brk2_cmp9 enable

This bit enables the tim_brk2_cmp9 for the timer's tim_brk2 input. tim_brk2_cmp9 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp9 input disabled

1: tim_brk2_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

0000:tim_ocref_clr0

0001:tim_ocref_clr1

...

1111: tim_ocref_clr15

Refer to *Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BK2INP**: TIM_BKIN2 input polarity

This bit selects the TIM_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIM_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIM_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BK2INE**: TIM_BKIN2 input enable

This bit enables the TIM_BKIN2 alternate function input for the timer's tim_brk2 input. TIM_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIM_BKIN2 input disabled

1: TIM_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Refer to Section 41.3.2: TIM1/TIM8/TIM20 pins and internal signals for product specific implementation.

41.6.29 TIM capture/compare register 7 (TIM_CCR7)

Address offset: 0x070

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR7[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR7[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR7[19:0]**: Capture/compare 7 value

CCR7 is the value to be loaded in the actual compare 7 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR4 register (bit OC7PE). Else the preload value is copied in the active compare 7 register when an update event occurs.

The active compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc7 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR7[15:0]. The CCR7[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR7[19:4]. The CCR7[3:0] bitfield contains the dithered part.

41.6.30 TIM capture/compare mode register 4 (TIM_CCMR4)

Address offset: 0x078

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channel 7 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7M[4:3]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7CE	OC7M[2:0]			OC7PE	OC7FE	Res.	Res.
								rw	rw	rw	rw	rw	rw		

Bits 31:18 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC7CE:** Output compare 7 clear enable

0: tim_oc7ref is not affected by the tim_ocref_clr_int signal

1: tim_oc7ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 41.3.23: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC7M[4:0]**: Output compare 7 mode

These bits define the behavior of the output reference signal tim_ocref7. tim_oc7ref is active high whereas tim_oc7 active level depends on CC7P bit.

00000:Frozen - The comparison between the output compare register TIM_CCR7 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001:Set channel 7 to active level on match. tim_oc7ref signal is forced high when the counter TIM_CNT matches the capture/compare register 7 (TIM_CCR7).

00010:Set channel 7 to inactive level on match. tim_oc7ref signal is forced low when the counter TIM_CNT matches the capture/compare register 7 (TIM_CCR7).

00011:Toggle - tim_oc7ref toggles when TIM_CNT = TIM_CCR7.

00100:Force inactive level - tim_oc7ref is forced low.

00101:Force active level - tim_oc7ref is forced high.

00110:PWM mode 1 - In upcounting, channel 7 is active as long as TIM_CNT<TIM_CCR7 else inactive. In downcounting, channel 7 is inactive (tim_oc7ref = 0) as long as TIM_CNT>TIM_CCR7 else active (tim_oc5ref = 1).

00111:PWM mode 2 - In upcounting, channel 7 is inactive as long as TIM_CNT<TIM_CCR7 else active. In downcounting, channel 7 is active as long as TIM_CNT>TIM_CCR7 else inactive.

01000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC7PE**: Output compare 7 preload enable

0: Preload register on TIM_CCR7 disabled. TIM_CCR7 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR7 enabled. Read/Write operations access the preload register. TIM_CCR7 preload value is loaded in the active register at each update event.

These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC7FE**: Output compare 7 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC7 behaves normally depending on counter and CCR7 values even when the trigger is ON. The minimum delay to activate CC7 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC7 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC7 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 Reserved, must be kept at reset value.

41.6.31 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BK4NM[1:0]		BK4M[1:0]		BK3NM[1:0]		BK3M[1:0]		BK2NM[1:0]		BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **BK4NM[1:0]**: Break mode for tim_oc4n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 13:12 **BK4M[1:0]**: Break mode for tim_oc4 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 11:10 **BK3NM[1:0]**: Break mode for tim_oc3n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **BK3M[1:0]**: Break mode for tim_oc3 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 7:6 **BK2NM[1:0]**: Break mode for tim_oc2n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 5:4 **BK2M[1:0]**: Break mode for tim_oc2 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

41.6.32 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for t_{DTG} selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK2[6:0] if DBK2[7] = 0
- DBK2[5:0] if DBK2[7:6] = 10
- DBK2[4:0] if DBK2[7:5] = 110 or 111

DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: Due to internal resynchronization of the break event on tim_ker_ck, the effective delayed break time is the programmed Delay2 plus up to two additional timer clock cycles.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for t_{DTG} selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK1[6:0] if DBK1[7] = 0
- DBK1[5:0] if DBK1[7:6] = 10
- DBK1[4:0] if DBK1[7:5] = 110 or 111

DBK1[7:5]=0xx => Delay1=DBK1[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK1[7:5]=10x => Delay1=(64+DBK1[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK1[7:5]=110 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK1[7:5]=111 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: Due to internal resynchronization of the break event on tim_ker_ck, the effective delayed break time is the programmed Delay1 plus up to two additional timer clock cycles.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

41.6.33 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	OOS4N	OOS4	OOS3N	OOS3	OOS2N	OOS2	OOS1N	OOS1
			rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control

This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.

0: tim_ocx and tim_ocxn outputs are in the default idle state

1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits

Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **OOS4N**: Output override state for output tim_oc4n

This bit defines the state for output tim_oc4n when OOC = 1

0: tim_oc4n = 0

1: tim_oc4n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **OOS4**: Output override state for output tim_oc4

This bit defines the state for output tim_oc4 when OOC = 1

0: tim_oc4 = 0

1: tim_oc4 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **OOS3N**: Output override state for output tim_oc3n

This bit defines the state for output tim_oc3n when OOC = 1

0: tim_oc3n = 0

1: tim_oc3n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **OOS3**: Output override state for output tim_oc3

This bit defines the state for output tim_oc3 when OOC = 1

0: tim_oc3 = 0

1: tim_oc3 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **OOS2N**: Output override state for output tim_oc2n
This bit defines the state for output tim_oc2n when OOC = 1
0: tim_oc2n = 0
1: tim_oc2n = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **OOS2**: Output override state for output tim_oc2
This bit defines the state for output tim_oc2 when OOC = 1
0: tim_oc2 = 0
1: tim_oc2 = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **OOS1N**: Output override state for output tim_oc1n
This bit defines the state for output tim_oc1n when OOC = 1
0: tim_oc1n = 0
1: tim_oc1n = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1
This bit defines the state for output tim_oc1 when OOC = 1
0: tim_oc1 = 0
1: tim_oc1 = 1
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

41.6.34 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).

0000: Disabled
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers.

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11111: 32 transfers

Example: Let us consider the following transfer: DBL = 6 & DBA = TIM2_CR1.

–If DBL = 6 and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIM_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIM_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIM_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:

00000: TIM_CR1
00001: TIM_CR2
00010: TIM_SMCR
...

41.6.35 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

41.6.36 TIM1/TIM8/TIM20 register map

TIM1/TIM8/TIM20 registers are mapped as 16-bit addressable registers as described in the table below:

Table 474. TIM1/TIM8/TIM20 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIM_CR1	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	TGO2PSC[4:0]					Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0				0	0	0	0	0				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIM_CR2	TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	MMS2[4:0]				OIS7	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	TIM_SMCR	Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS[4:3]		ETP	ECE	ETPS [1:0]		ETF[3:0]			MSM	TS[2:0]			OCCS	SMS[2:0]			
	Reset value			0	0	0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	TIM_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	TIM_SR	TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	ODS	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	CC7IF	CC6IF	CC5IF	B2GF	BGF	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	TIM_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0

Table 474. TIM1/TIM8/TIM20 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	TIM_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIM_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]		OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR2	Res.	Res.	Res	Res.	Res.	Res.	Res	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIM_CCER	Res.	Res.	Res.	Res.	Res.	Res.	CC7P	CC7E	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value							0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	TIM_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIM_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	TIM_RCR	Res.	Res.	Res	Res.	Res.	Res.	Res	Res	Res.	Res.	Res	Res	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	TIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIM_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIM_CCR3	Res.	Res.	Res	Res.	Res.	Res.	Res	Res	Res.	Res.	Res	Res	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIM_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	TIM_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 474. TIM1/TIM8/TIM20 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	TIM_CCR5	GC5C3	GC5C2	GC5C1	GC5C4	GC5C4O	GC5C3O	GC5C2O	GC5C1O	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	TIM_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	TIM_CCMR3	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC5M[4:3]		OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value							0	0							0	0	0	0	0	0	0	0			0	0	0	0	0	0		
0x054	TIM_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	TIM_ECR	Res.	Res.	Res.	Res.	Res.	PWPR SC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	TIM_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIM_AF1	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	ETRSEL [4:0]						BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	TIM_AF2	BK2CMP20E	BK2CMP19E	BK2CMP18E	BK2CMP17E	BK2CMP16E	BK2CMP15E	BK2CMP14E	BK2CMP13E	BK2CMP12E	BK2CMP11E	BK2CMP10E	BK2CMP9E	OCRSEL[3:0]				Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x06C	Reserved	Res.																															
0x070	TIM_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR719:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	Reserved	Res.																															
0x078	TIM_CCMR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC7CE	OC7M [2:0]			OC7PE	OC7FE	Res.	Res.
	Reset value															0	0									0	0	0	0	0	0		
0x07C.. 0x8C	Reserved	Res.																															
0x090	TIM_MPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK4NM[1:0]		BK4M[1:0]		BK3NM[1:0]		BK3M[1:0]		BK2NM[1:0]		BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 474. TIM1/TIM8/TIM20 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x094	TIM_MPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x098.. 0x09C	Reserved	Res.																															
0x0A0	TIM_OOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	OOS4N	OOS4	OOS3N	OOS3	OOS2N	OOS2	OOS1N	OOS1
	Reset value																				0					0	0	0	0	0	0	0	0
0x0A4.. 0x3D8	Reserved	Res.																															
0x3DC	TIM_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIM_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

42 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

42.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 42.4.27: Timer synchronization*.

42.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to four independent channels for:
  - Input capture.
  - Output compare.
  - PWM generation (edge- and center-aligned modes).
  - One-pulse mode output.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  - Input capture.
  - Output compare.
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
- Trigger input for external clock or cycle-by-cycle current management.
- ADC synchronization for jitter-free sampling points.

42.3 TIM2/TIM3/TIM4/TIM5 implementation

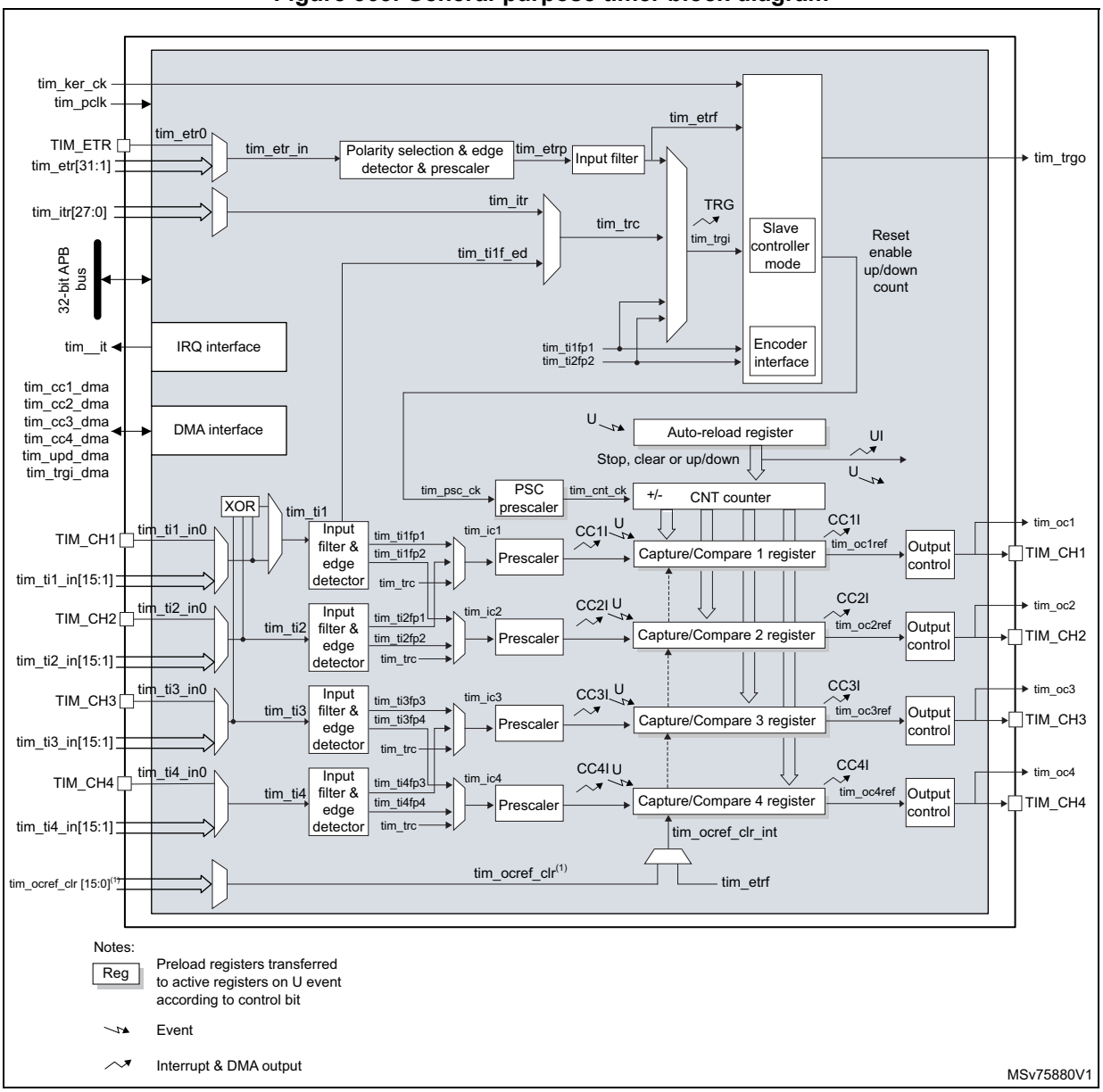
Table 475. STM32G56x/55x/54x/53x general purpose timers

Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	32-bit	32-bit	32-bit
OCREF clear selection	Yes	Yes	Yes	Yes
Sources	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr	tim_etrf tim_ocref_clr

42.4 TIM2/TIM3/TIM4/TIM5 functional description

42.4.1 Block diagram

Figure 563. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 42.3: TIM2/TIM3/TIM4/TIM5 implementation*.

42.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 476 and *Table 477* in this section summarize the TIM inputs and outputs.

Table 476. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 477. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[31:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 477. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture/compare [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.
tim_ocx (x = 1 to 4)	Output	Internal output compare signals of channels 1 to 4. these outputs can be used for triggering/synchronizing other internal peripherals.

Tables below list the sources connected to the tim_ti[4:1] input multiplexers.

Table 478. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1	TIM5_CH1
tim_ti1_in1	comp1_out	comp1_out	comp1_out	comp1_out
tim_ti1_in2	comp3_out	comp3_out	comp3_out	comp3_out
tim_ti1_in3	comp5_out	comp5_out	comp5_out	comp5_out
tim_ti1_in4	comp7_out	comp7_out	comp7_out	comp7_out
tim_ti1_in5	-	-	-	-
tim_ti1_in6	-	-	-	-
tim_ti1_in7	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_ti1_in8	tim15_trgo	tim2_trgo	tim2_trgo	tim2_trgo
tim_ti1_in9	play1_out15	play1_out8	play1_out6	-
tim_ti1_in10	fdcan1_rxeof_evt	fdcan2_rxeof_evt	fdcan3_rxeof_evt	fdcan1_rxeof_evt
tim_ti1_in[15:11]	-	-	-	-

Table 479. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2	TIM5_CH2
tim_ti2_in1	comp2_out	comp2_out	comp2_out	comp2_out

Table 479. Interconnect to the tim_ti2 input multiplexer (continued)

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in2	comp4_out	comp4_out	comp4_out	comp4_out
tim_ti2_in3	comp6_out	comp6_out	comp6_out	comp6_out
tim_ti2_in4	comp8_out	comp8_out	comp8_out	comp8_out
tim_ti2_in5	-	-	-	-
tim_ti2_in6	-	-	-	-
tim_ti2_in7	tim3_trgo	tim15_trgo	tim3_trgo	tim3_trgo
tim_ti2_in8	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_ti2_in9	play1_out15	play1_out9	play1_out11	-
tim_ti2_in10	fdcan1_txeof_evt	fdcan2_txeof_evt	fdcan3_txeof_evt	fdcan1_txeof_evt
tim_ti2_in[15:11]	-	-	-	-

Table 480. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3	TIM5_CH3
tim_ti3_in1	tim4_trgo	tim4_trgo	tim15_trgo	tim4_trgo
tim_ti3_in2	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo
tim_ti3_in3	play1_out3	play1_out10	play1_out12	-
tim_ti3_in4	fdcan2_rxeof_evt	fdcan3_rxeof_evt	fdcan1_rxeof_evt	fdcan2_rxeof_evt
tim_ti3_in[15:5]	-	-	-	-

Table 481. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4	TIM5_CH4
tim_ti4_in1	tim5_trgo	tim5_trgo	tim5_trgo	tim15_trgo
tim_ti4_in2	-	-	-	-
tim_ti4_in3	play1_out0	play1_out11	-	-
tim_ti4_in4	fdcan2_txeof_evt	fdcan3_txeof_evt	fdcan1_txeof_evt	fdcan2_txeof_evt
tim_ti4_in[15:5]	-	-	-	-

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 482. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	tim1_trgo2	tim1_trgo2	tim1_trgo2	tim1_trgo2
tim_itr2	-	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr3	tim3_trgo	-	tim3_trgo	tim3_trgo
tim_itr4	tim4_trgo	tim4_trgo	-	tim4_trgo
tim_itr5	tim5_trgo	tim5_trgo	tim5_trgo	-
tim_itr6	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_itr7	tim8_trgo2	tim8_trgo2	tim8_trgo2	tim8_trgo2
tim_itr8	tim9_trgo	tim9_trgo	tim9_trgo	tim9_trgo
tim_itr9	tim10_oc1	tim10_oc1	tim10_oc1	tim10_oc1
tim_itr10	tim11_oc1	tim11_oc1	tim11_oc1	tim11_oc1
tim_itr11	tim12_trgo	tim12_trgo	tim12_trgo	tim12_trgo
tim_itr12	tim13_oc1	tim13_oc1	tim13_oc1	tim13_oc1
tim_itr13	tim14_oc1	tim14_oc1	tim14_oc1	tim14_oc1
tim_itr14	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo
tim_itr15	tim16_oc1	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr16	tim17_oc1	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr17	tim20_trgo	tim20_trgo	tim20_trgo	tim20_trgo
tim_itr18	tim20_trgo2	tim20_trgo2	tim20_trgo2	tim20_trgo2
tim_itr19	-	-	-	-
tim_itr20	-	-	-	-
tim_itr21	hrtim1_out_sync2	hrtim1_out_sync2	hrtim1_out_sync2	hrtim1_out_sync2
tim_itr[27:22]	-	-	-	-

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 483. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	comp1_out	comp1_out	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out	comp2_out	comp2_out
tim_etr3	comp3_out	comp3_out	comp3_out	comp3_out
tim_etr4	comp4_out	comp4_out	comp4_out	comp4_out

Table 483. Interconnect to the tim_etr input multiplexer (continued)

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr5	comp5_out	comp5_out	comp5_out	comp5_out
tim_etr6	comp6_out	comp6_out	comp6_out	comp6_out
tim_etr7	comp7_out	comp7_out	comp7_out	comp7_out
tim_etr8	comp8_out	comp8_out	comp8_out	comp8_out
tim_etr9	-	-	-	-
tim_etr10	-	-	-	-
tim_etr11	-	-	-	-
tim_etr12	-	-	-	-
tim_etr13	-	TIM2_ETR	TIM2_ETR	TIM2_ETR
tim_etr14	TIM3_ETR	-	TIM3_ETR	TIM3_ETR
tim_etr15	TIM4_ETR	TIM4_ETR	-	TIM4_ETR
tim_etr16	TIM5_ETR	TIM5_ETR	TIM5_ETR	-
tim_etr17	adc1_awd1	adc2_awd1	adc3_awd1	-
tim_etr18	adc1_awd2	adc2_awd2	adc3_awd2	-
tim_etr19	adc1_awd3	adc2_awd3	adc3_awd3	-
tim_etr20	usb_sof	usb_sof	usb_sof	usb_sof
tim_etr21	-	-	-	-
tim_etr22	lse_ck	-	-	-
tim_etr23	play1_out4	play1_out4	play1_out5	-
tim_etr24	play1_out2	play1_out13	play1_out15	-
tim_etr[31:25]	-	-	-	-

Table 484. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr0	comp1_out	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out	comp2_out
tim_ocref_clr2	comp3_out	comp3_out	comp3_out	comp3_out
tim_ocref_clr3	comp4_out	comp4_out	comp4_out	comp4_out
tim_ocref_clr4	comp5_out	comp5_out	comp5_out	comp5_out
tim_ocref_clr5	comp6_out	comp6_out	comp6_out	comp6_out
tim_ocref_clr6	comp7_out	comp7_out	comp7_out	comp7_out
tim_ocref_clr7	comp8_out	comp8_out	comp8_out	comp8_out

Table 484. Interconnect to the tim_ocref_clr input multiplexer (continued)

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr[11:8]	-	-	-	-
tim_ocref_clr12	play1_out15	play1_out3	play1_out3	-
tim_ocref_clr13	play1_out7	play1_out2	play1_out1	-
tim_ocref_clr[15:14]	-	-	-	-

42.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIM_PSC

register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 564 and *Figure 565* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 564. Counter timing diagram with prescaler division change from 1 to 2

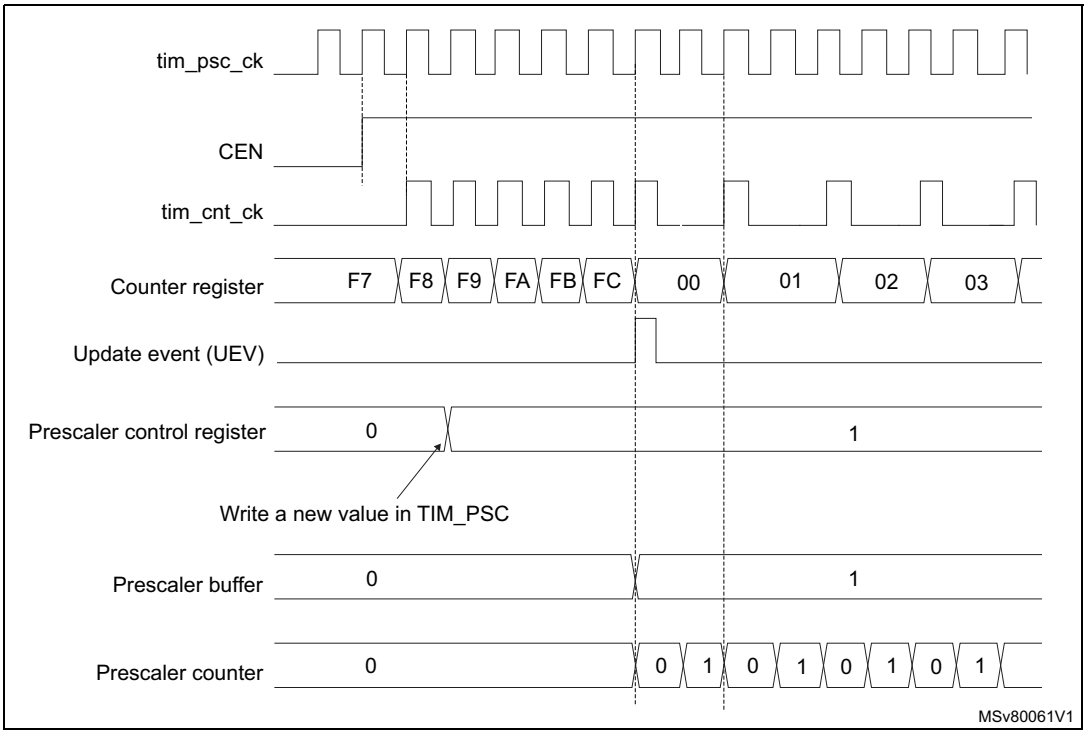
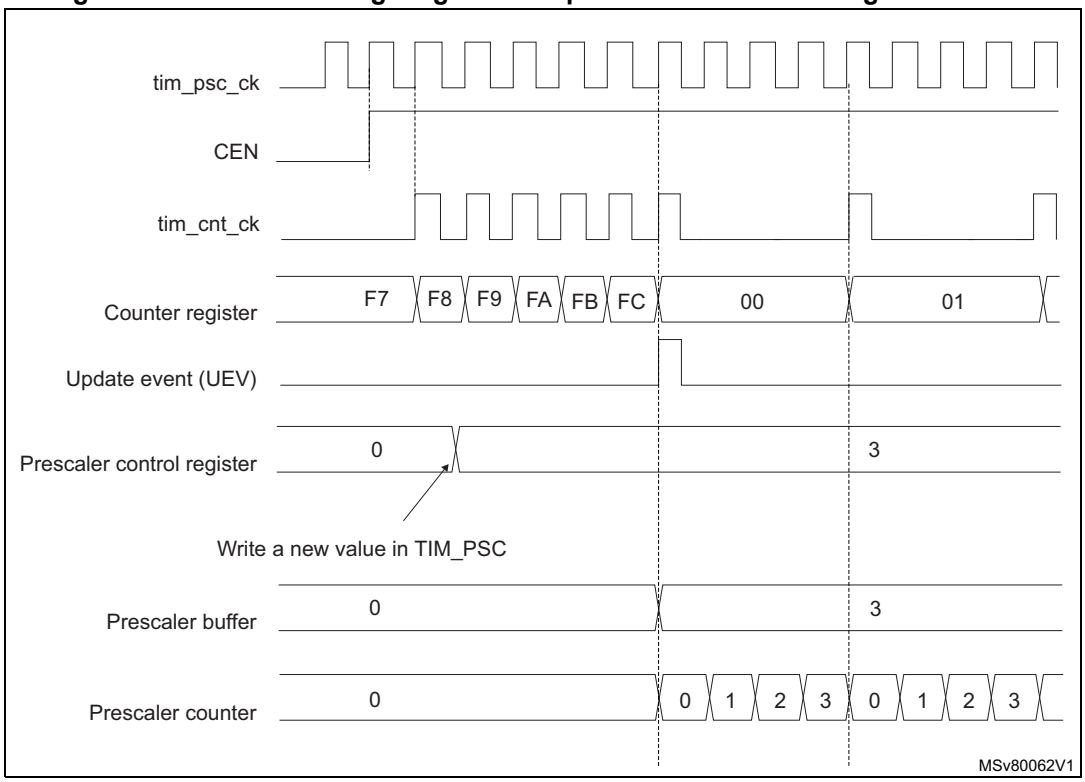


Figure 565. Counter timing diagram with prescaler division change from 1 to 4



42.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload shadow register is updated with the preload value (TIM_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 566. Counter timing diagram, internal clock divided by 1

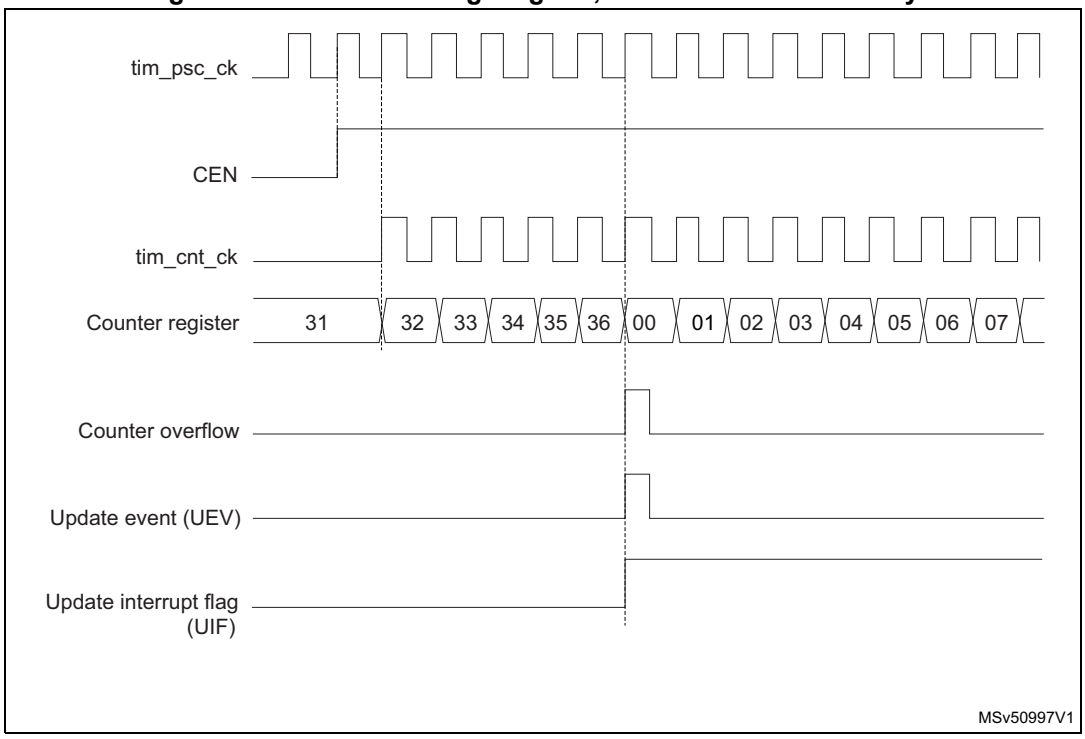


Figure 567. Counter timing diagram, internal clock divided by 2

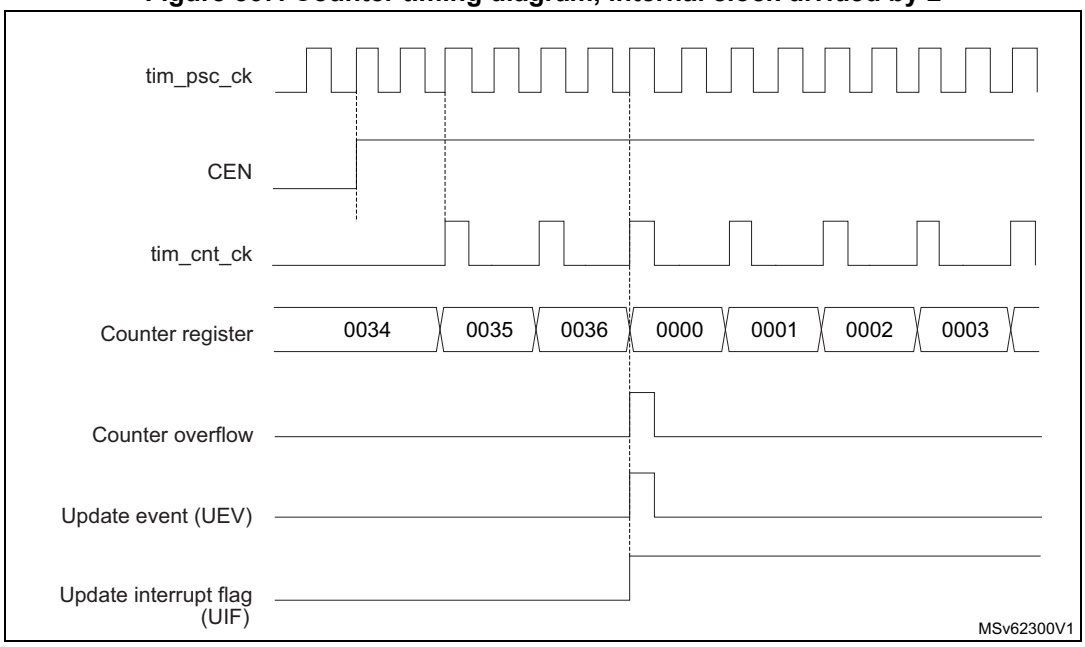


Figure 568. Counter timing diagram, internal clock divided by 4

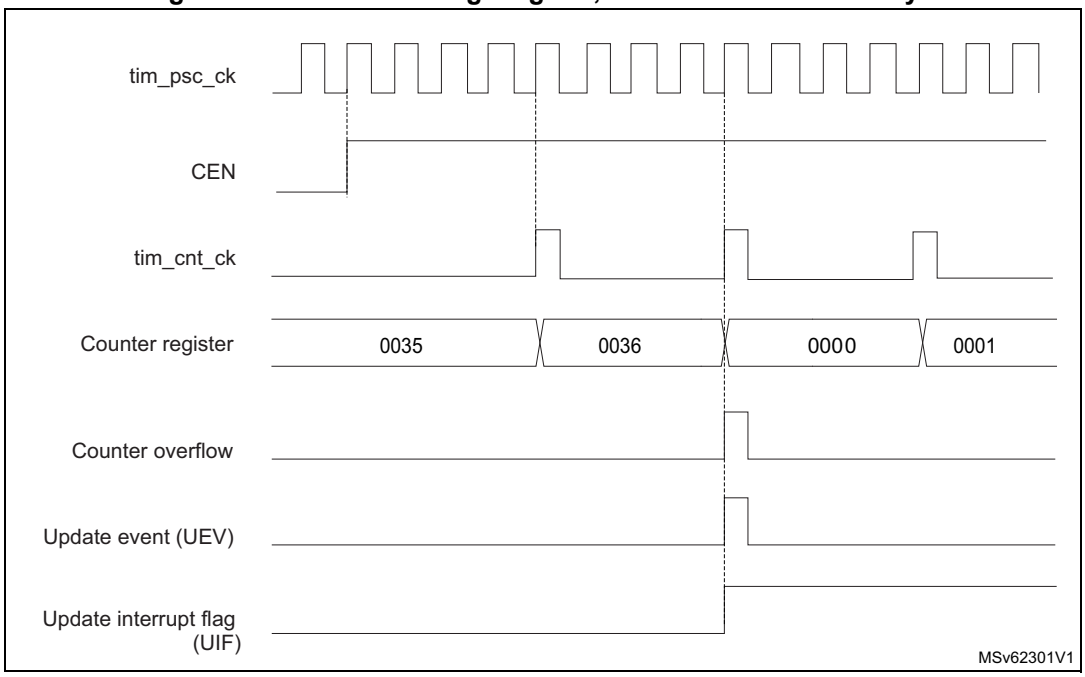


Figure 569. Counter timing diagram, internal clock divided by N

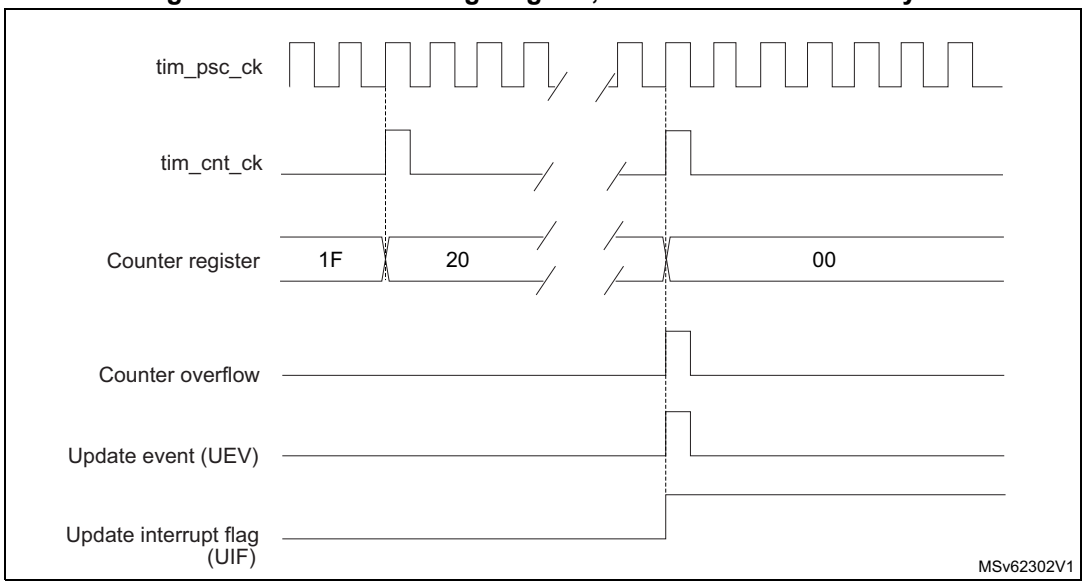


Figure 570. Counter timing diagram, Update event when ARPE = 0 (TIM_ARR not preloaded)

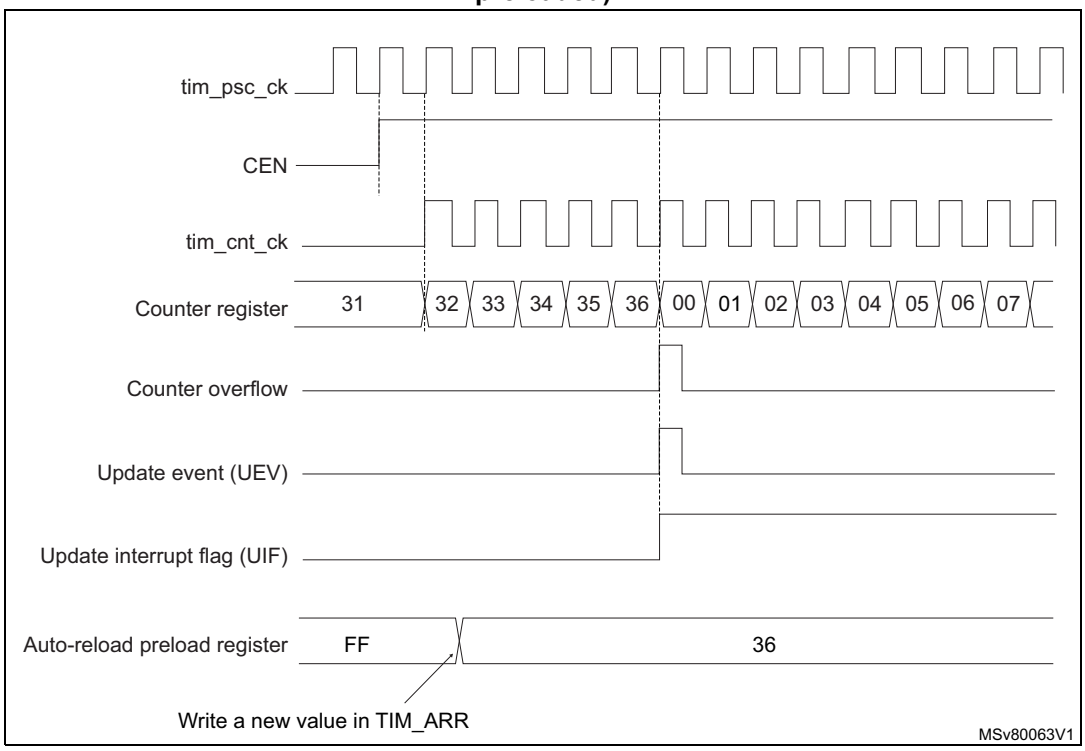
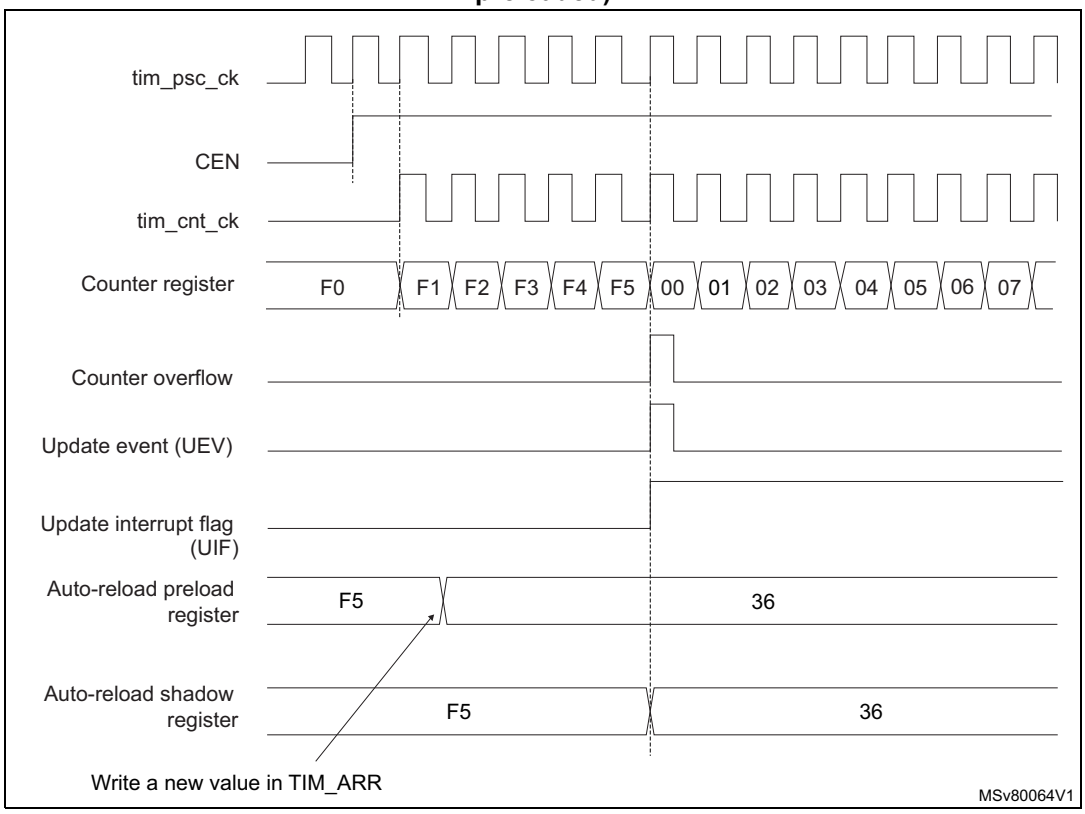


Figure 571. Counter timing diagram, Update event when ARPE = 1 (TIM_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 572. Counter timing diagram, internal clock divided by 1

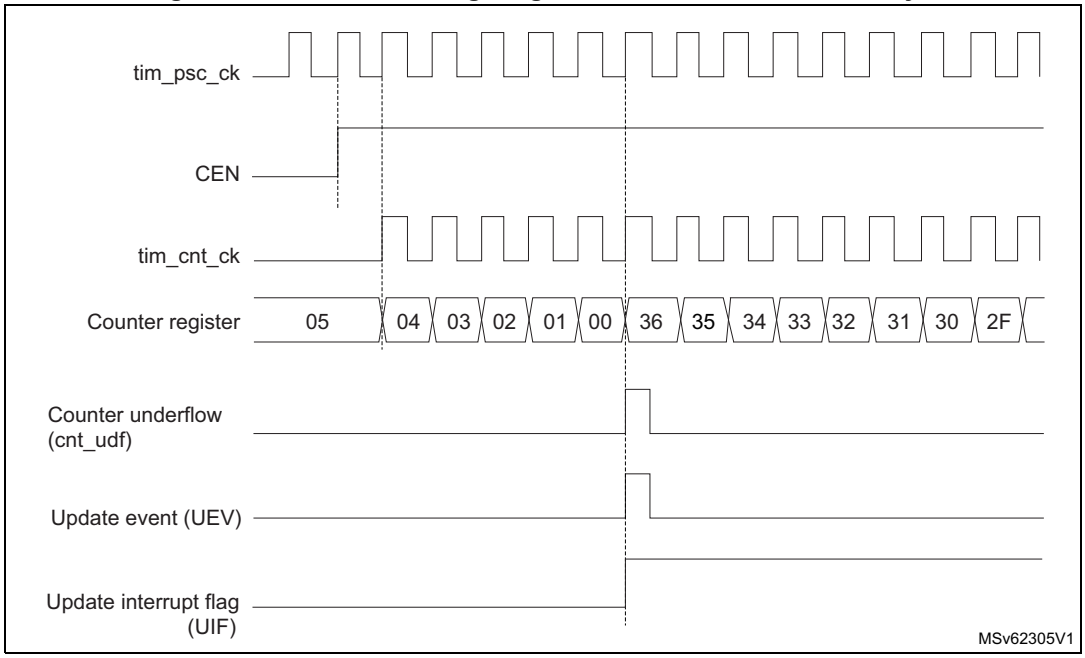


Figure 573. Counter timing diagram, internal clock divided by 2

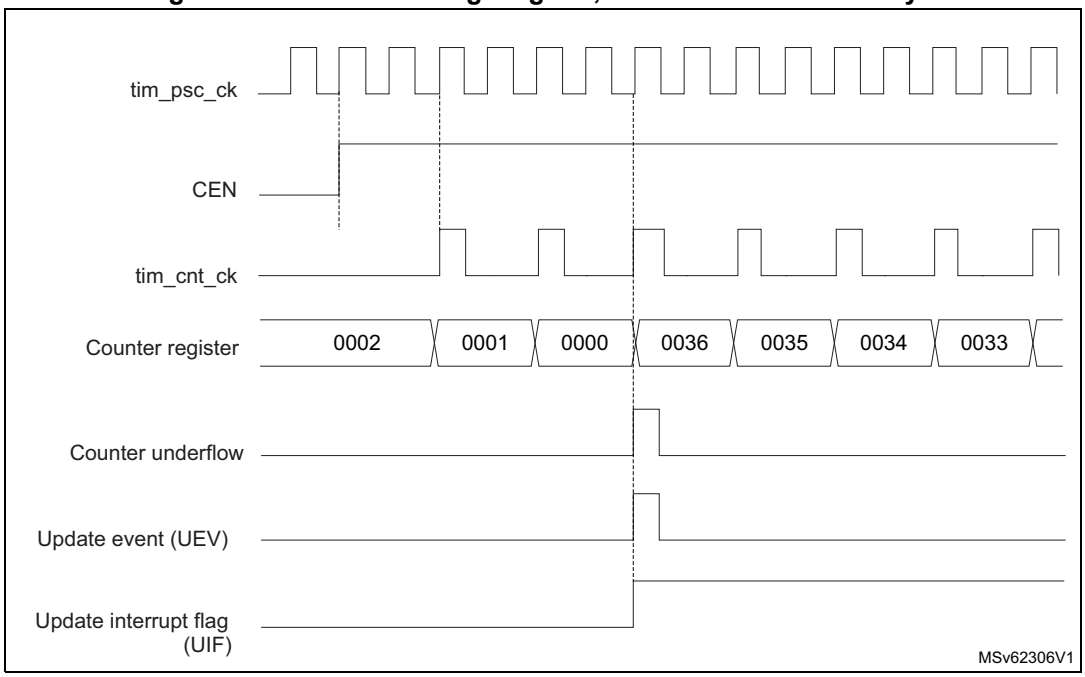


Figure 574. Counter timing diagram, internal clock divided by 4

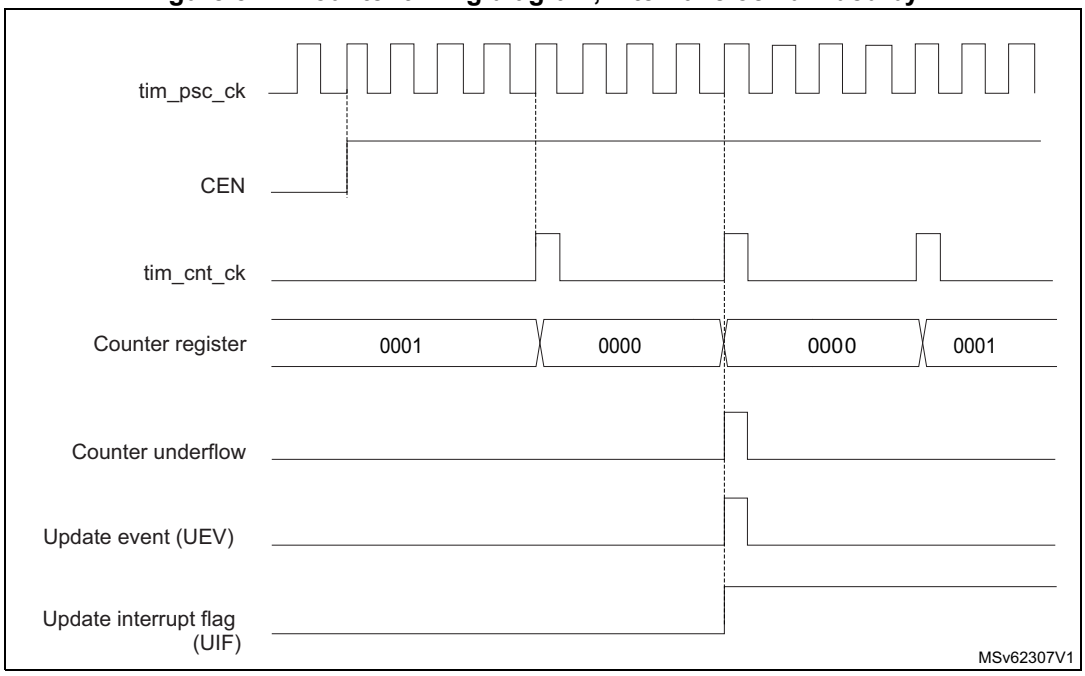


Figure 575. Counter timing diagram, internal clock divided by N

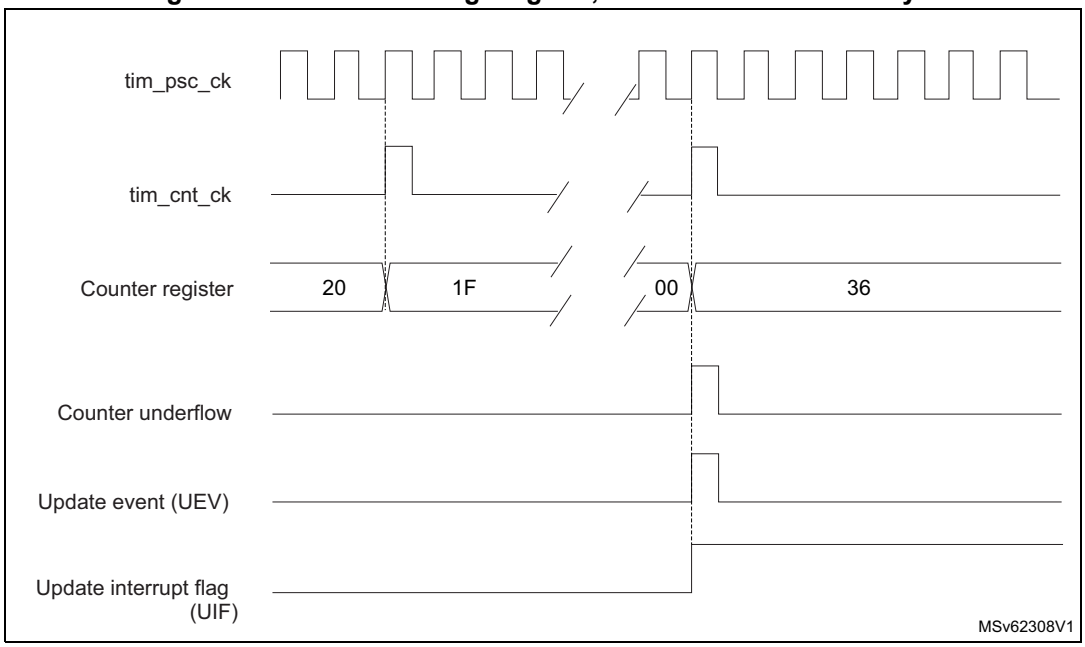
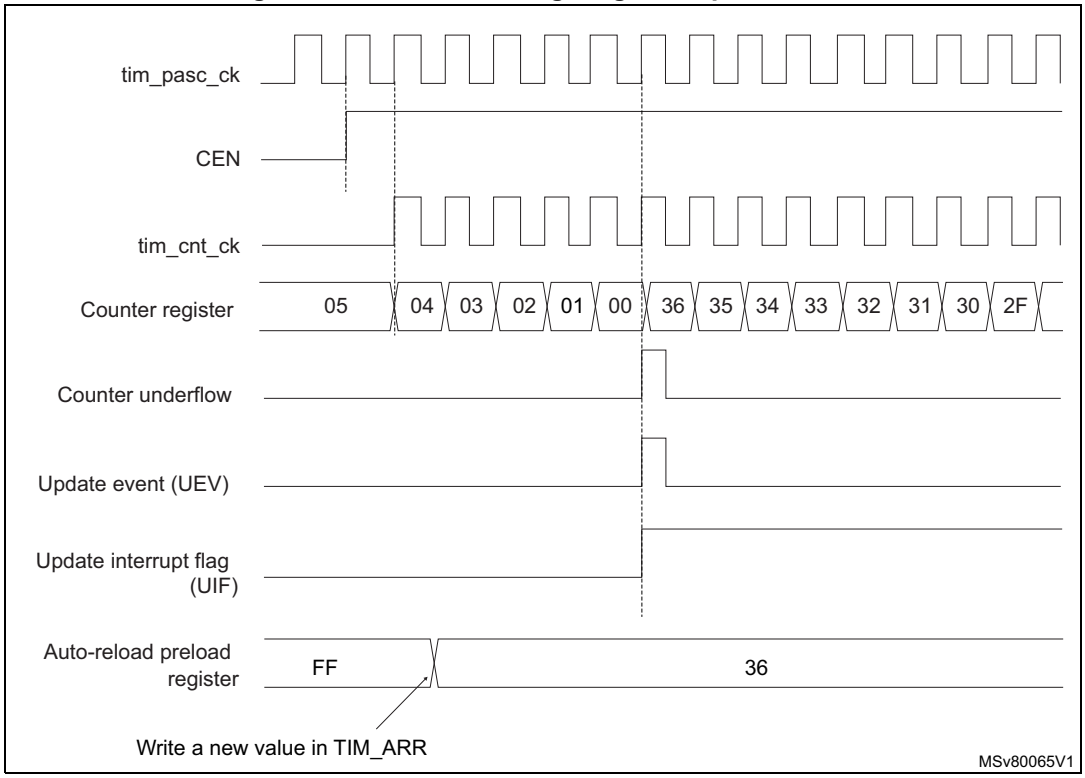


Figure 576. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (center aligned mode 1, CMS = 01), the counter counts up (center aligned mode 2, CMS = 10) the counter counts up and down (center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIM_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

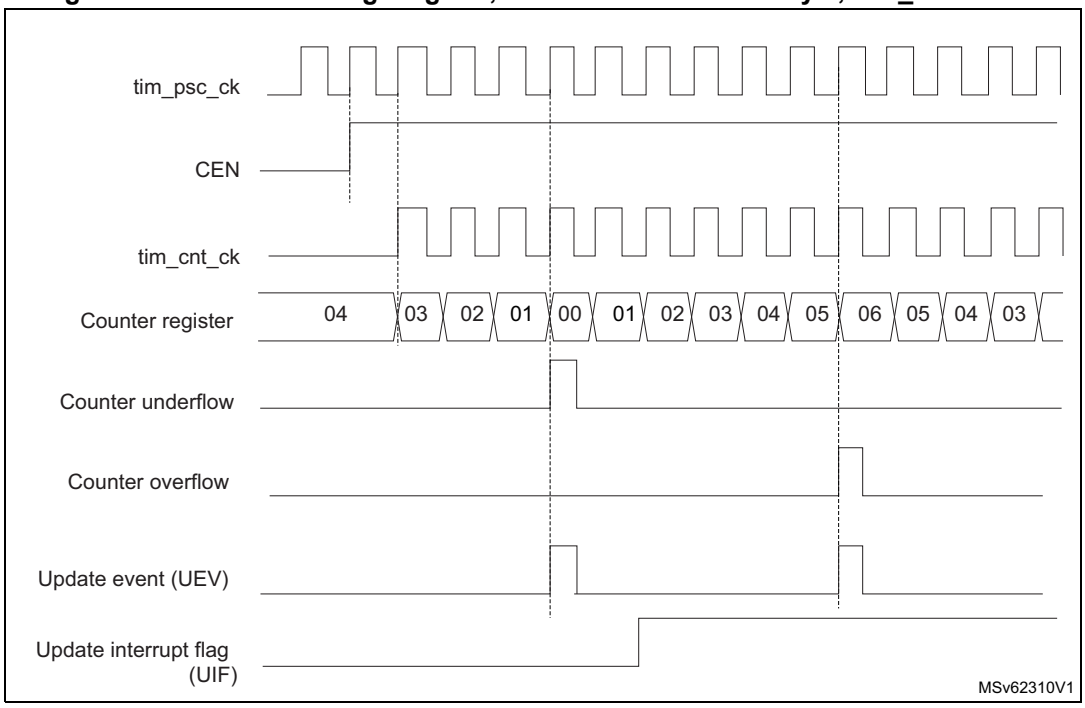
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 577. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 42.5.1: TIM control register 1 (TIM_CR1)*).

Figure 578. Counter timing diagram, internal clock divided by 2

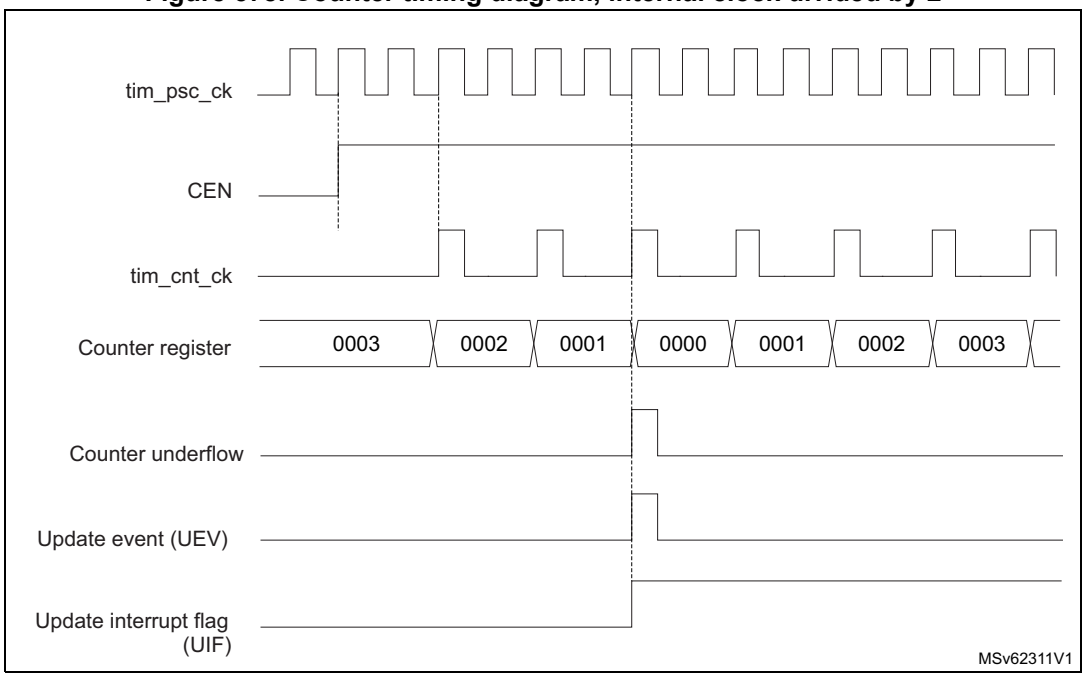
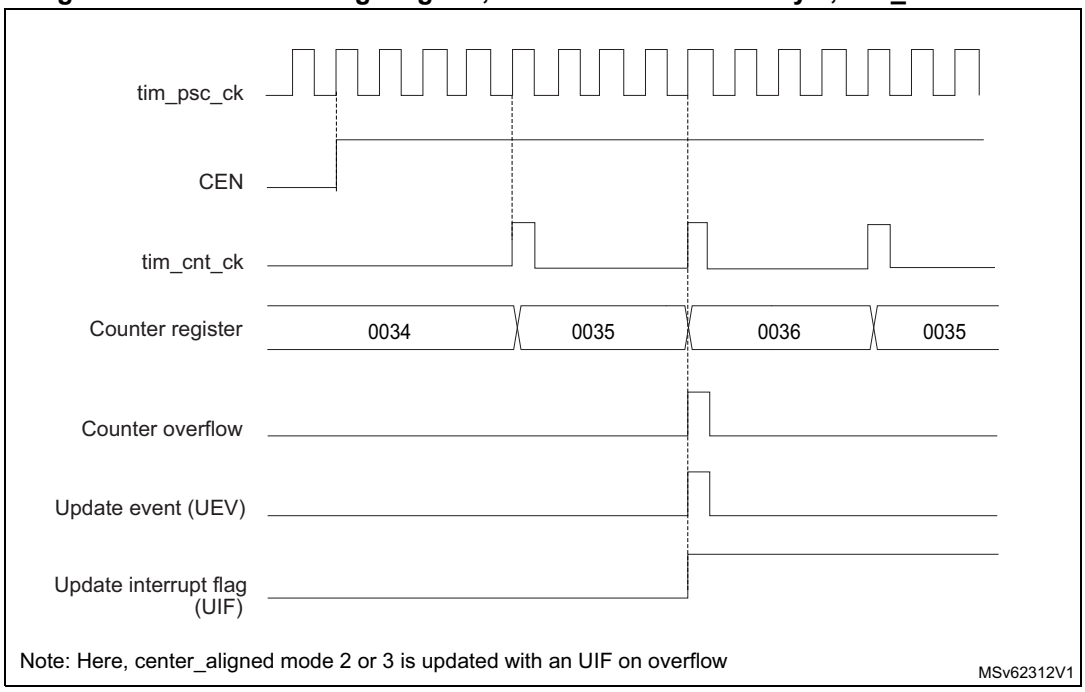


Figure 579. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 580. Counter timing diagram, internal clock divided by N

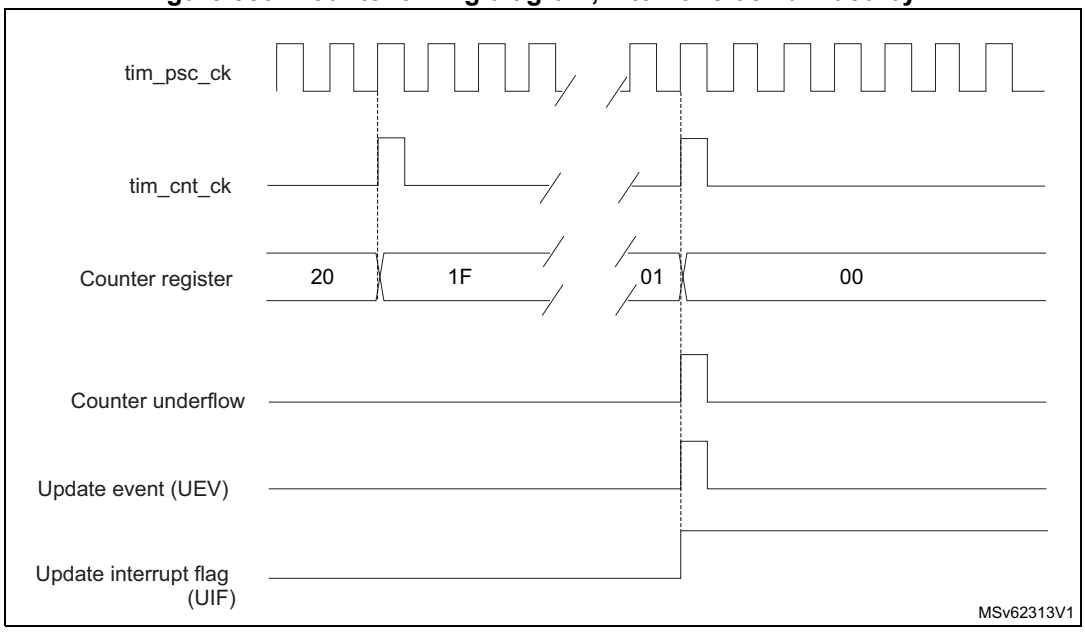


Figure 581. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

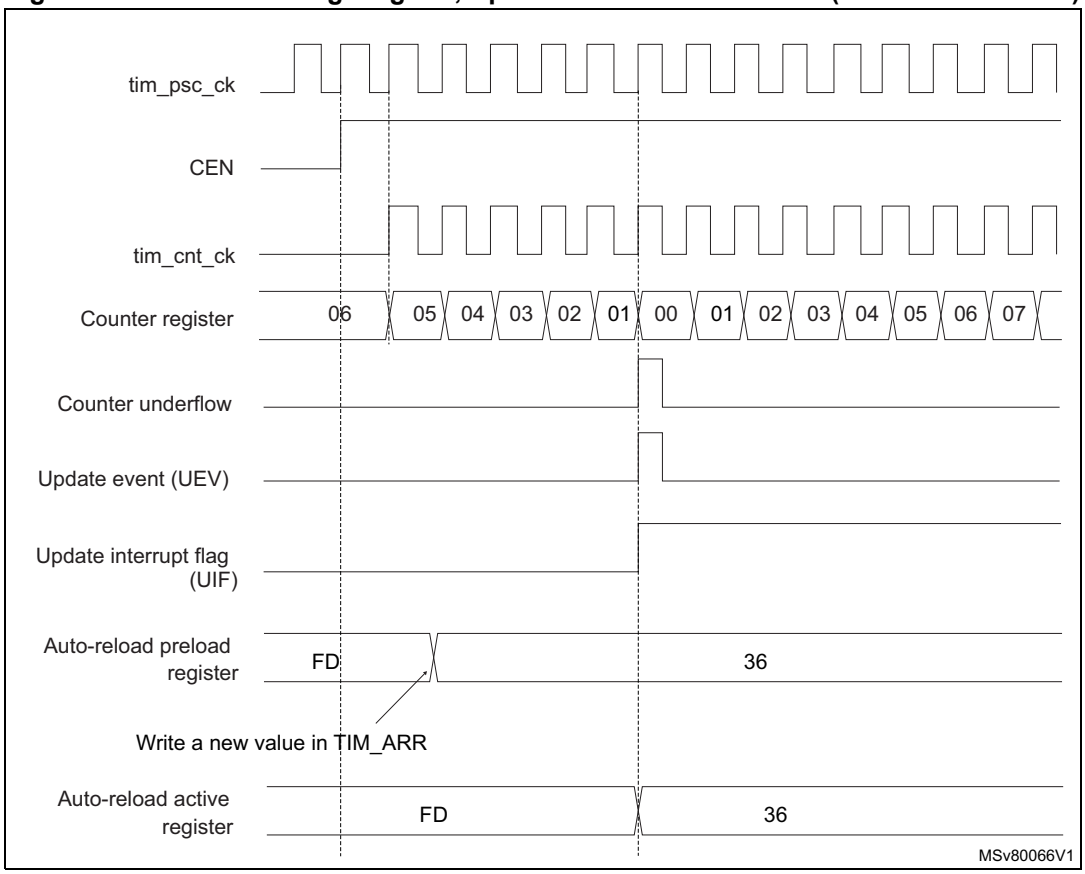
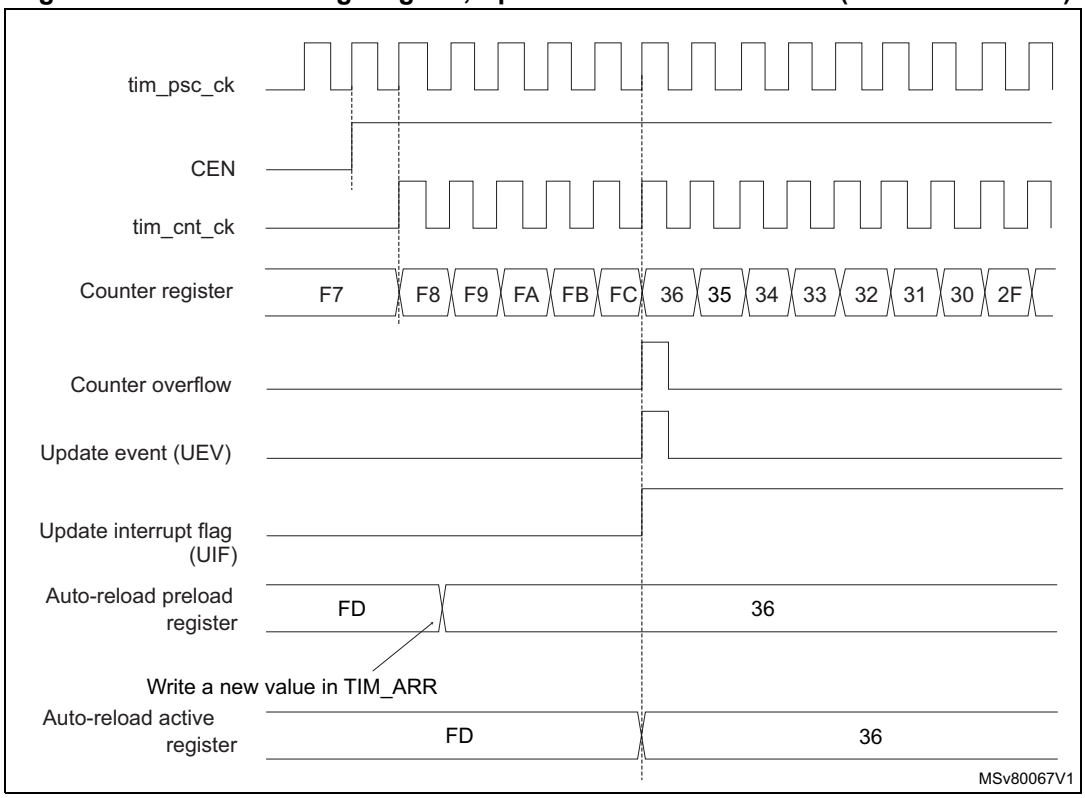


Figure 582. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



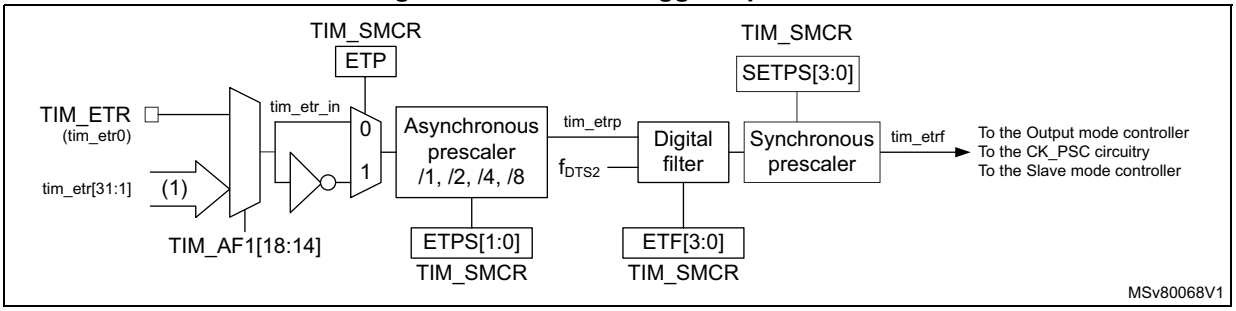
42.4.5 External trigger input

The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 42.4.6*)
- trigger for the slave mode (see *Section 42.4.27*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 42.4.16*)

Figure 583 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIM_SMCR register. The trigger frequency can be decreased with the prescalers programmed by the ETPS[1:0] and SETPS[3:0] bitfields and digitally filtered as per the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition the output (typically to reset a PWM output for a current limitation), and as a trigger for the slave mode controller.

Figure 583. External trigger input block



The external trigger input block features 2 prescalers with different functionalities:

- An asynchronous prescaler (division by 1, 2, 4 or 8), able to process tim_etr_in signals with frequencies above tim_ker_ck
- A linear synchronous prescaler (division by 1,2,3,..15), only able to operate with tim_etrp signal frequencies below tim_ker_ck/3

Note: When tim_etrf is used as PWM reset input for cycle-by-cycle current regulation (see Section 42.4.16), SETPS[3:0] must be set to 0 (no prescaling).

Note: The SETPS[3:0] bitfield is always preloaded: a new value will become active only once an update event has occured.

The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[4:0] bitfield in the TIM_AF1 register.

Refer to *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for the list of sources connected to the etr_in input in the product.

42.4.6 Clock selection

The counter clock can be provided by the following clock sources:

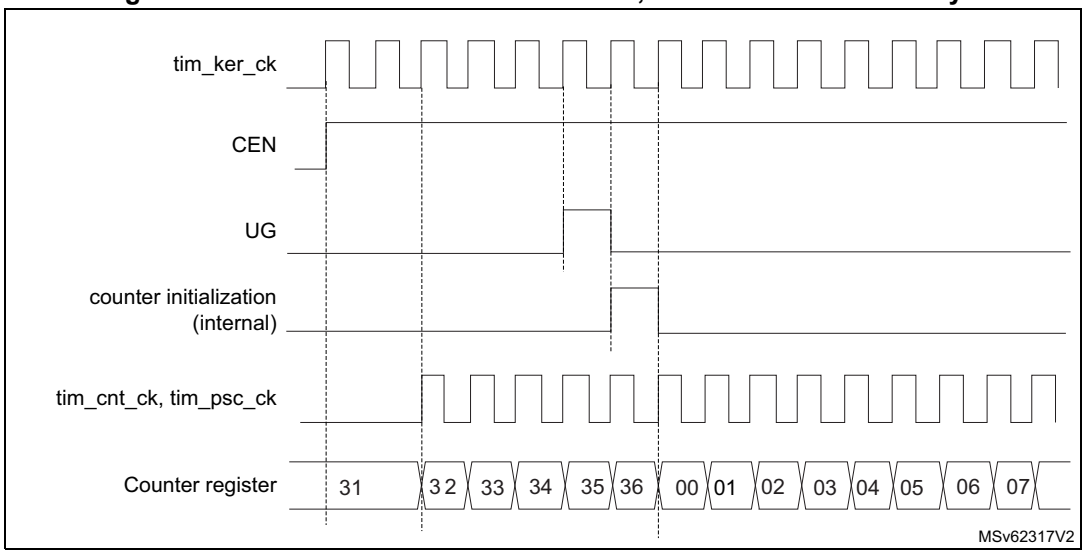
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000 in the TIM_SMCR register), then the CEN, DIR (in the TIM_CR1 register), and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 584 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

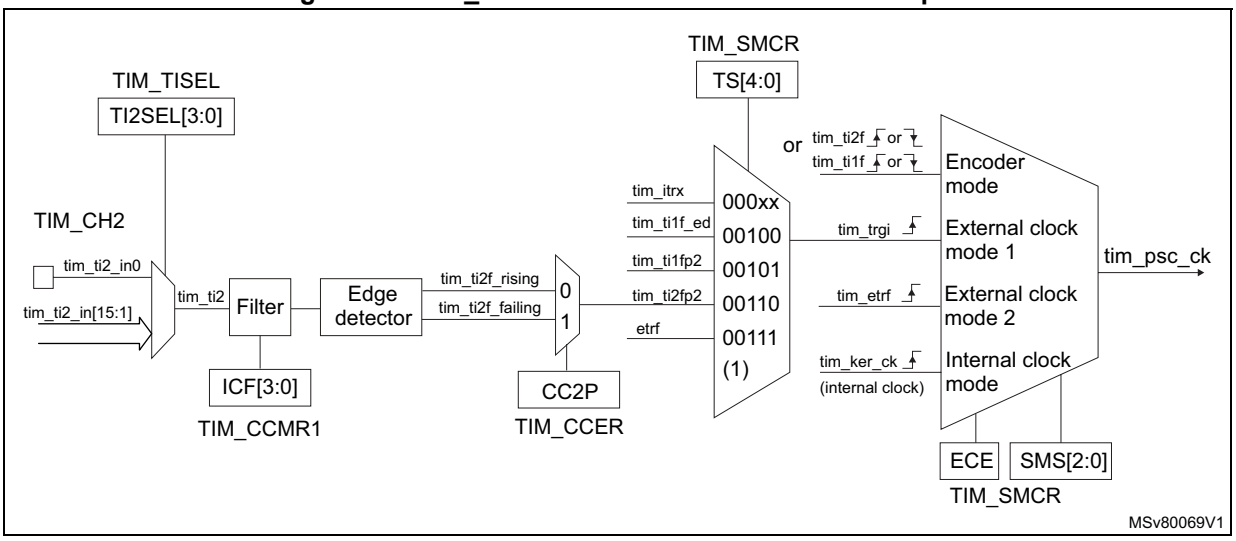
Figure 584. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 585. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).

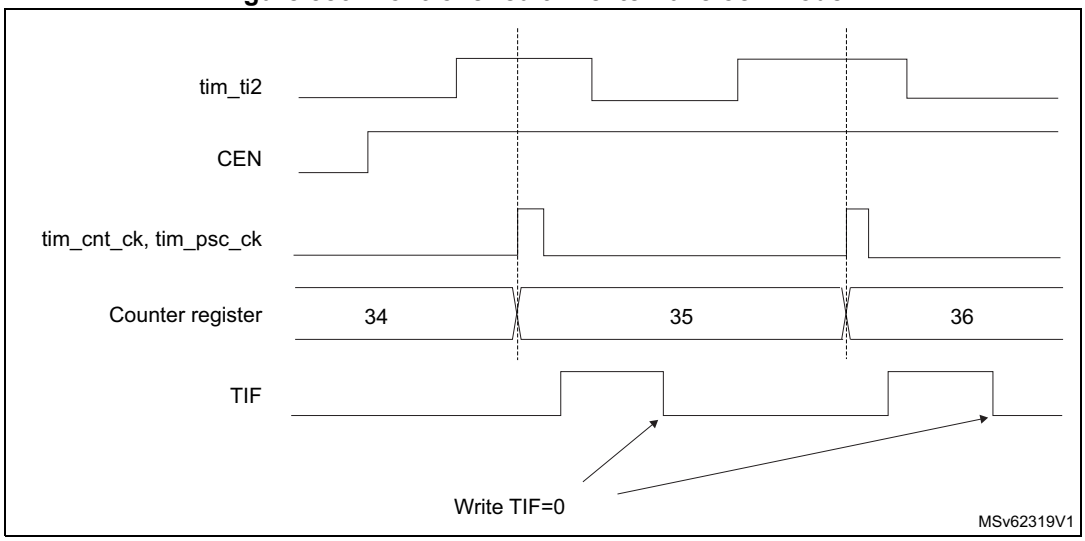
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 586. Control circuit in external clock mode 1



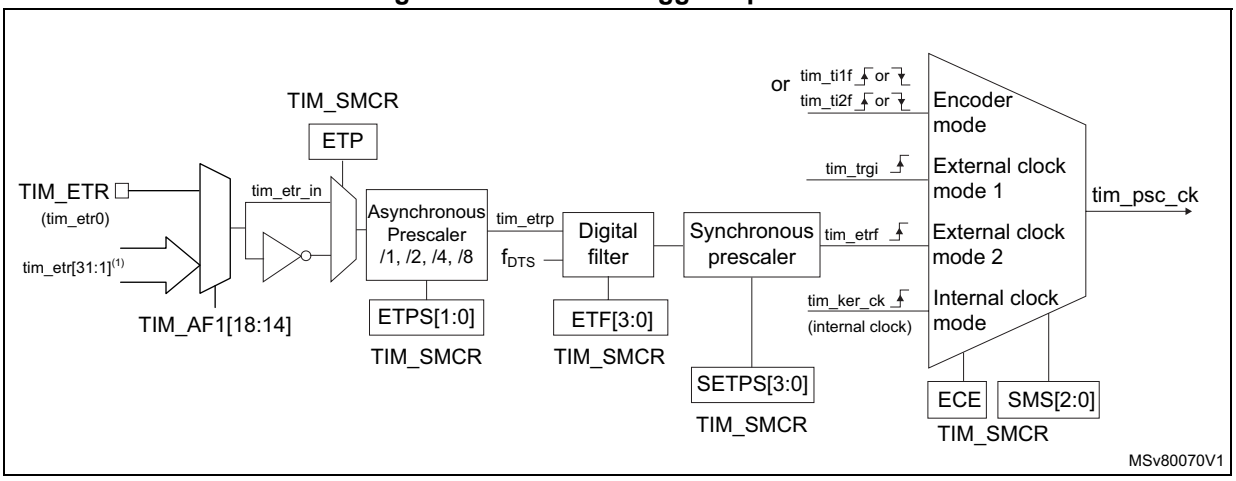
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIM_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 587 gives an overview of the external trigger input block.

Figure 587. External trigger input block



1. Refer to *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals*

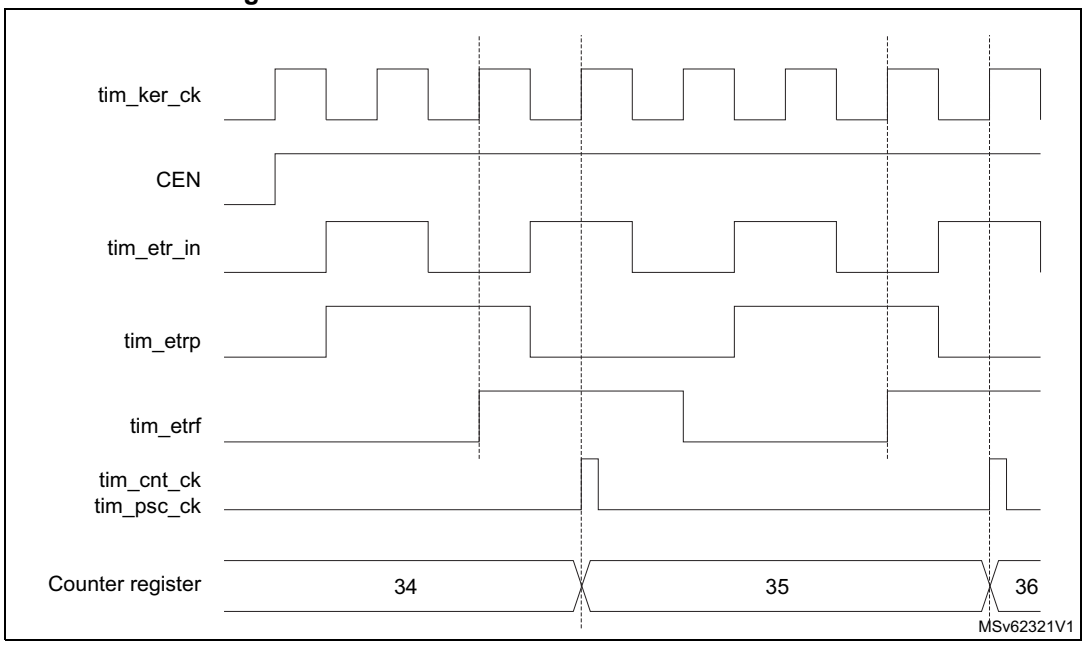
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[4:0] bits in the TIM_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIM_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 and SETPS[3:0] = 0 in the TIM_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP = 0 in the TIM_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIM_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

The counter counts once each two tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 588. Control circuit in external clock mode 2



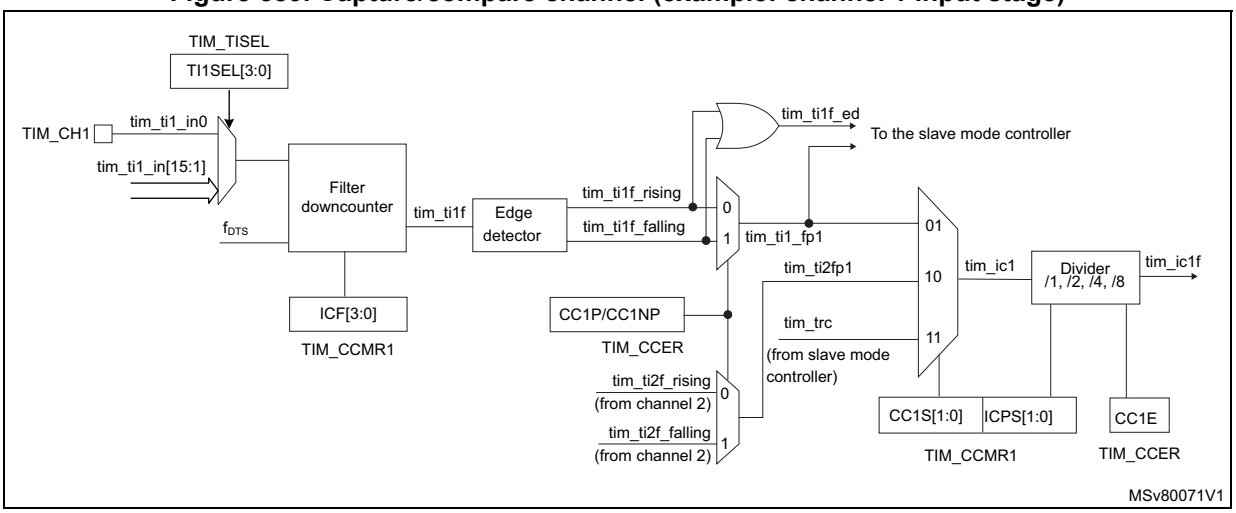
42.4.7 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 589. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 590. Capture/compare channel 1 main circuit

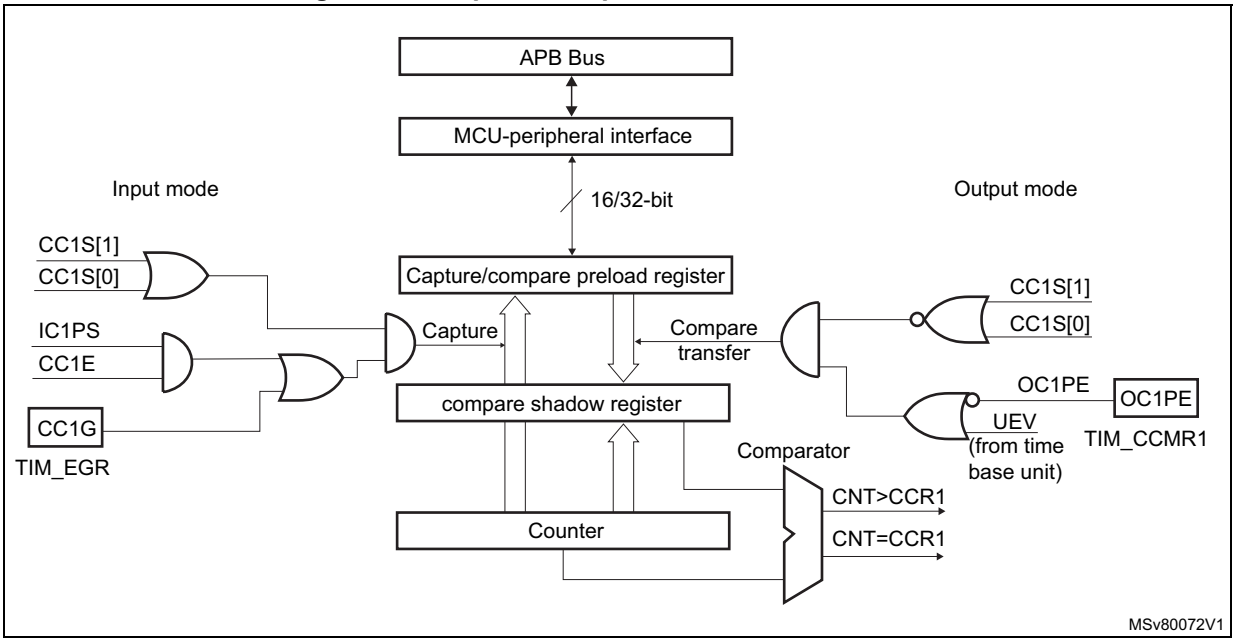
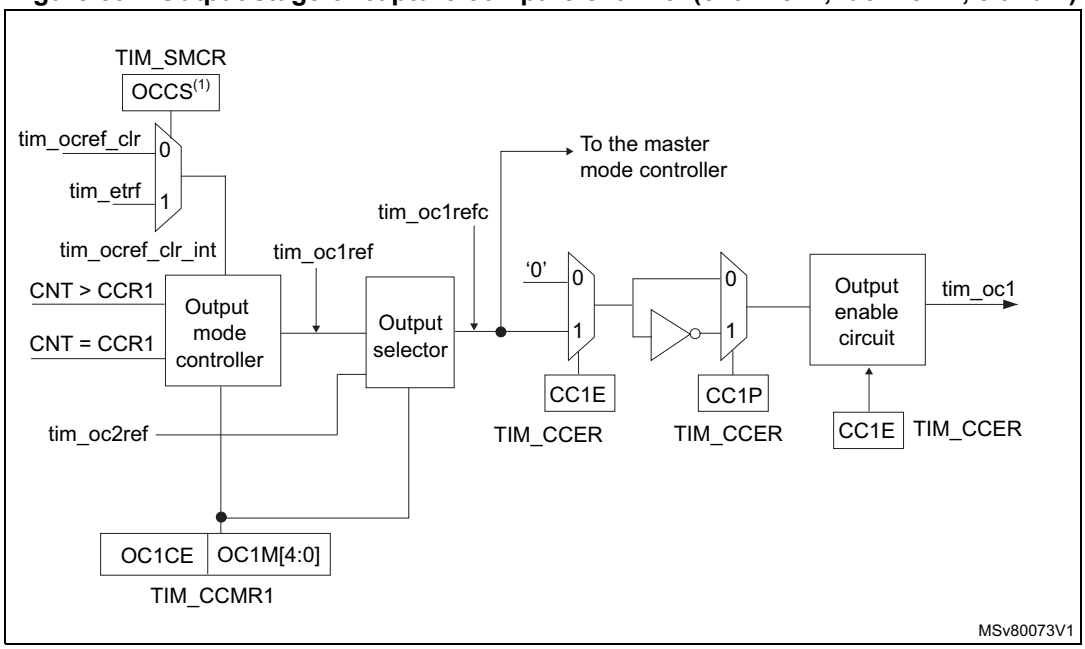


Figure 591. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

42.4.8 Input capture mode

In input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIM_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

42.4.9 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

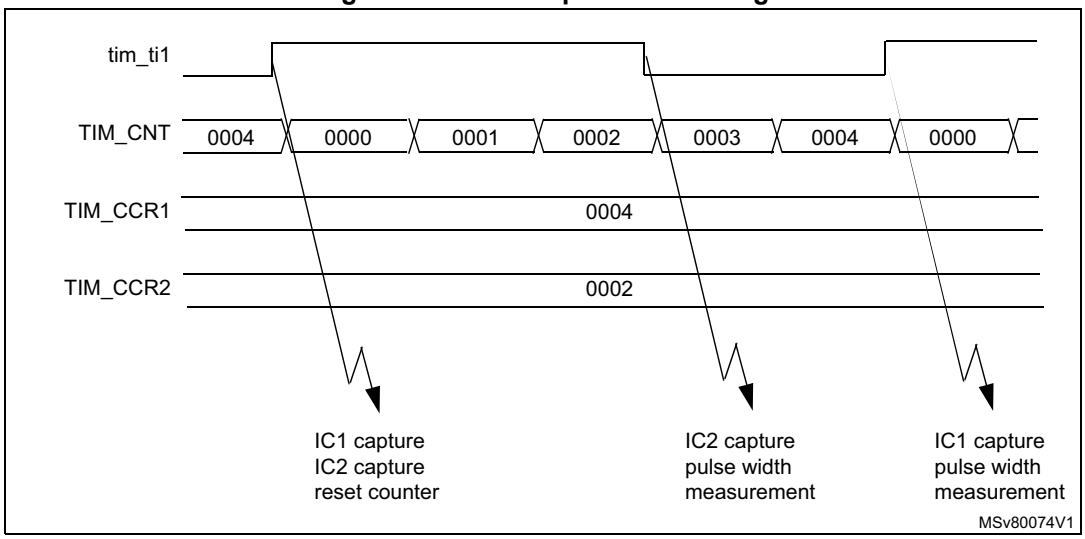
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 592. PWM input mode timing



1. The PWM input mode can be used only with the TIM_CH1/TIM_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

42.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, the user just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section.

42.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 00000), be

set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

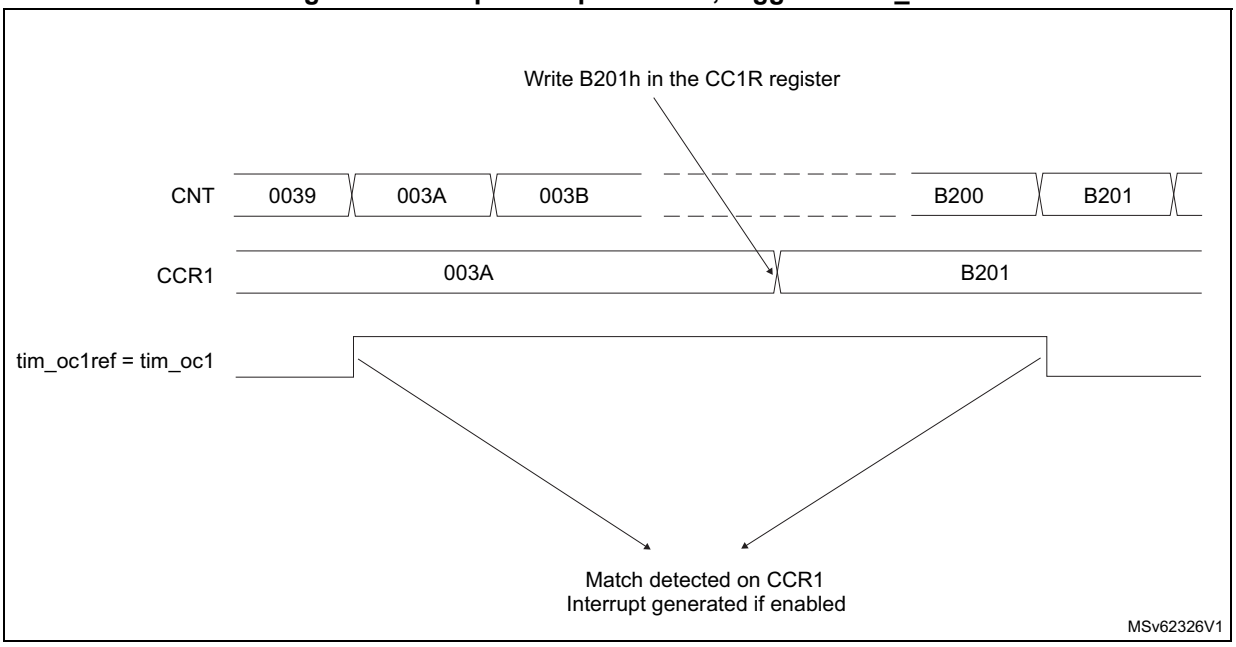
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in one-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 593*.

Figure 593. Output compare mode, toggle on tim_oc1



42.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIM_CCER register. Refer to the TIM_CCERx register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIM_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 00000) to one of the PWM modes (OCxM = 00110 or 00111). This forces the PWM by software while the timer is running.

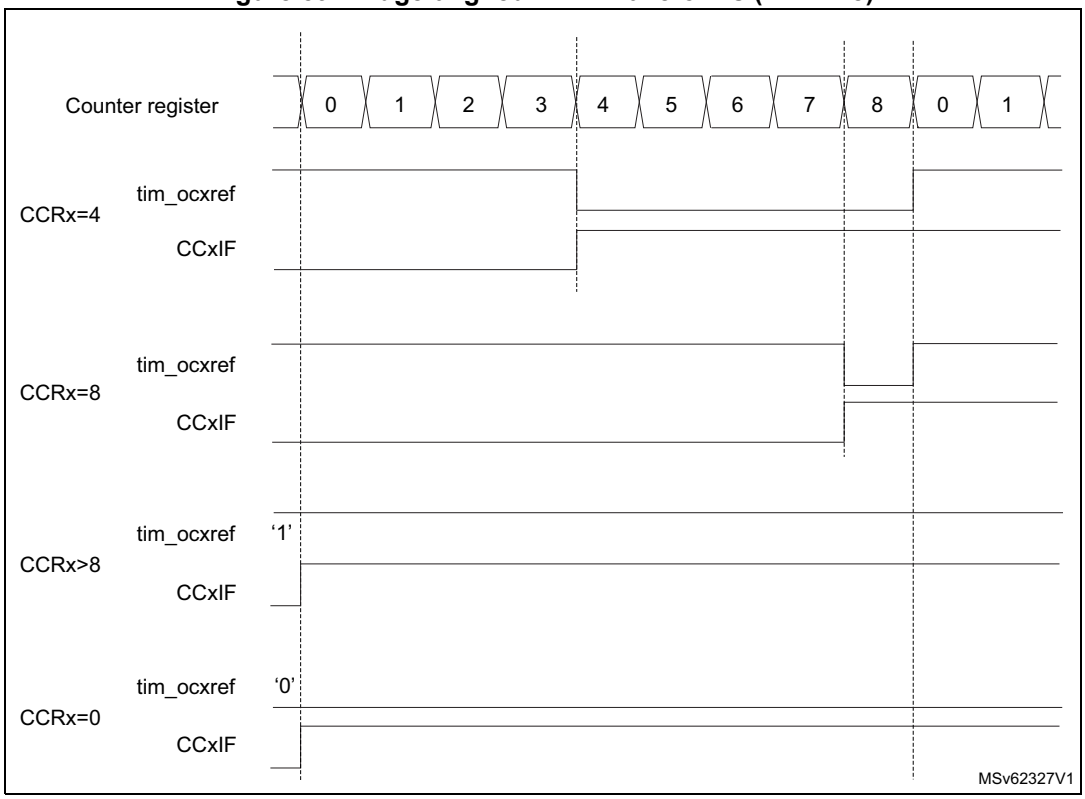
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Up-counting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT <TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 594* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 594. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIM_CR1 register is high. Refer to *Down-counting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT>TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

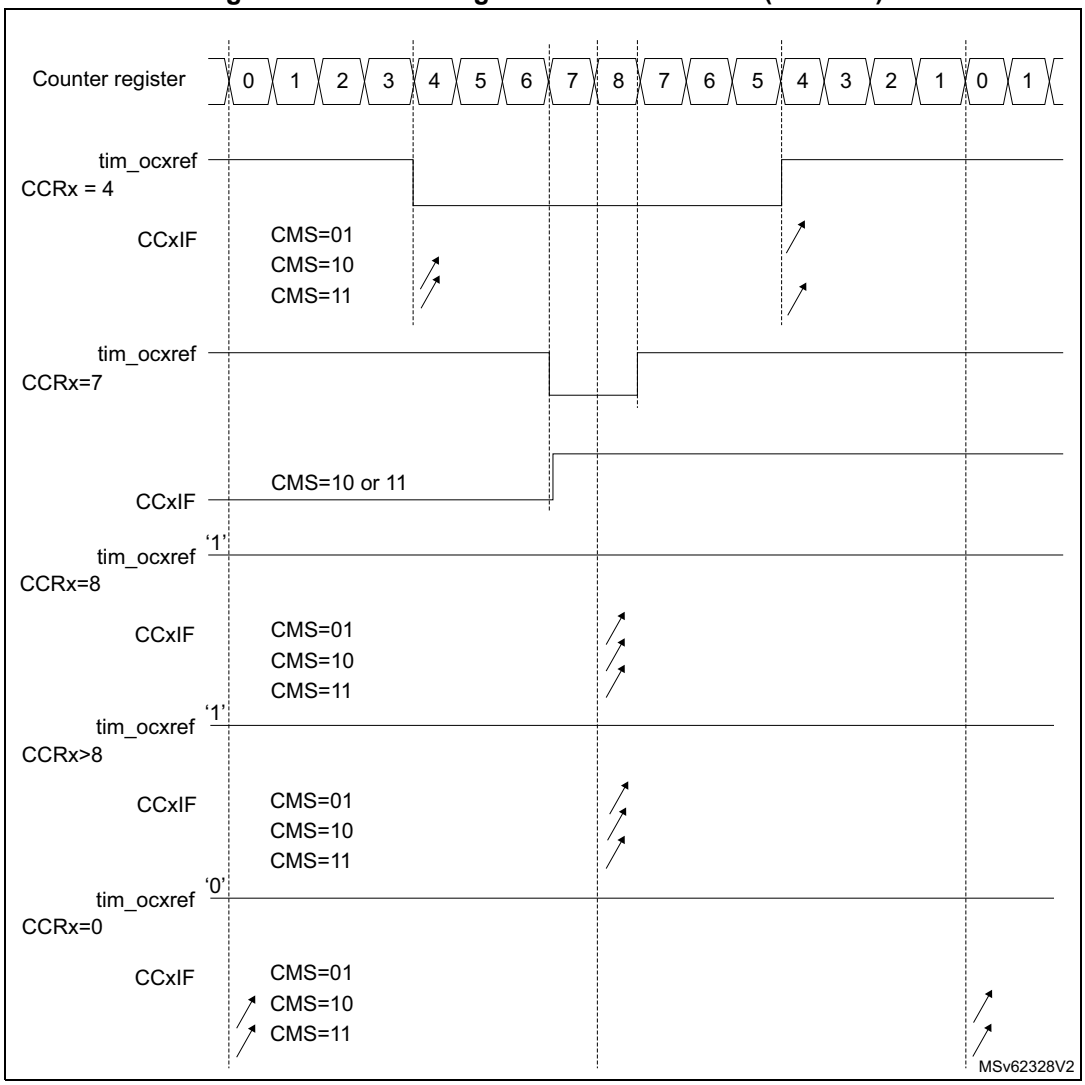
Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down-counting)*.

Figure 595 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 595. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

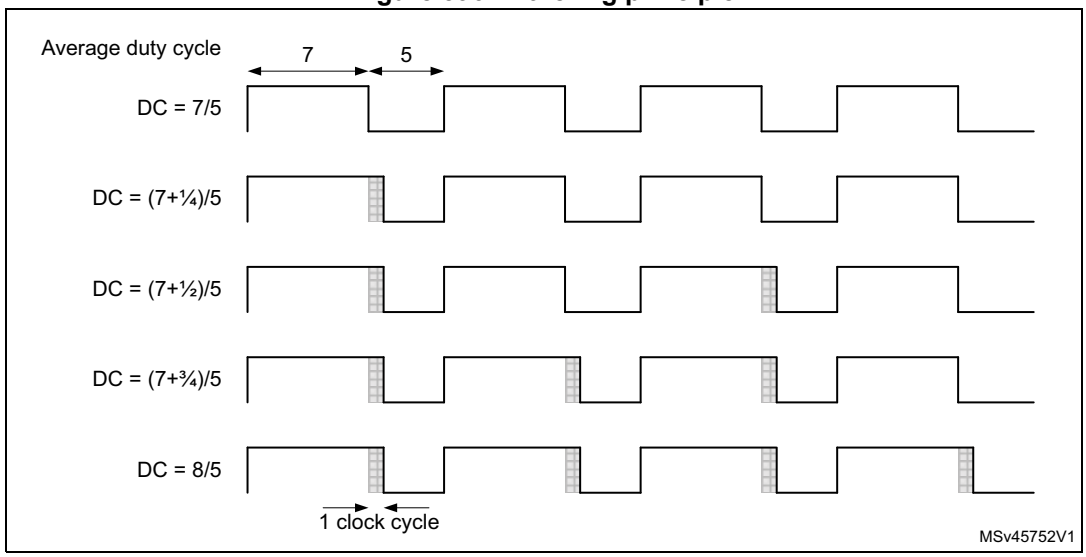
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT>TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 596* presents the dithering principle applied to four consecutive PWM cycles.

Figure 596. Dithering principle



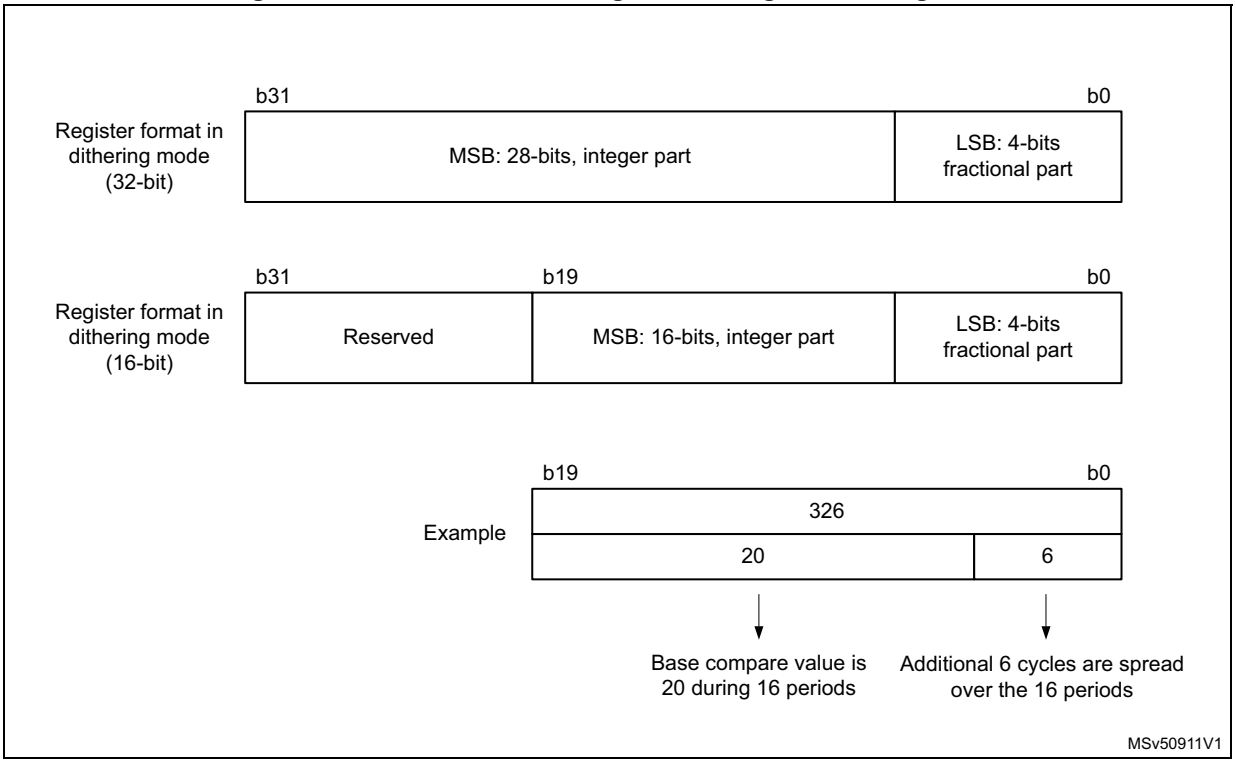
When the dithering mode is enabled, the register coding is changed as following (see *Figure 597* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 597. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIM_ARR and TIM_CCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on *Figure 598* and *Figure 599*, the dithering mode is used to increase the PWM resolution.

Figure 598. PWM resolution vs frequency (16-bit mode)

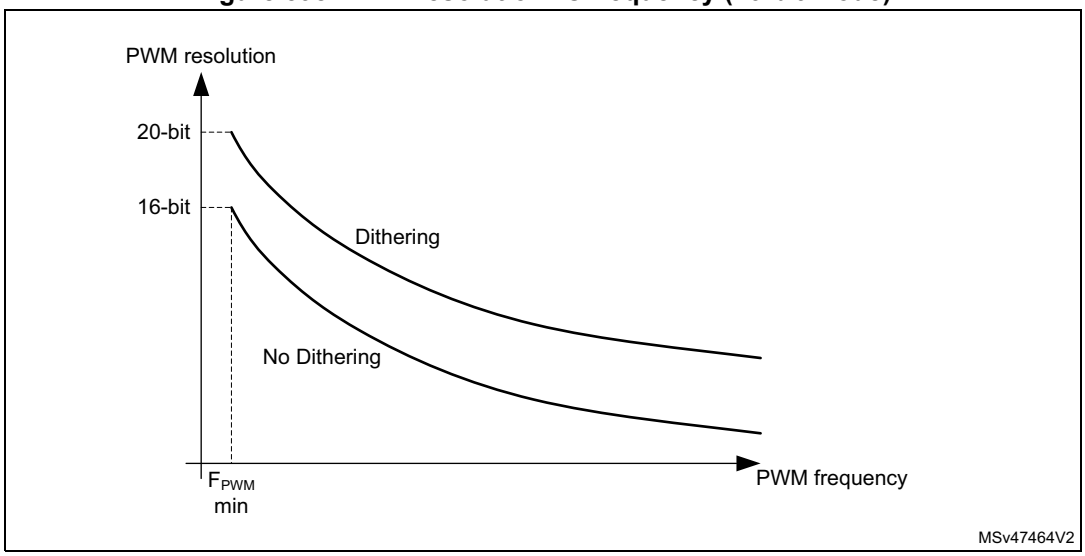
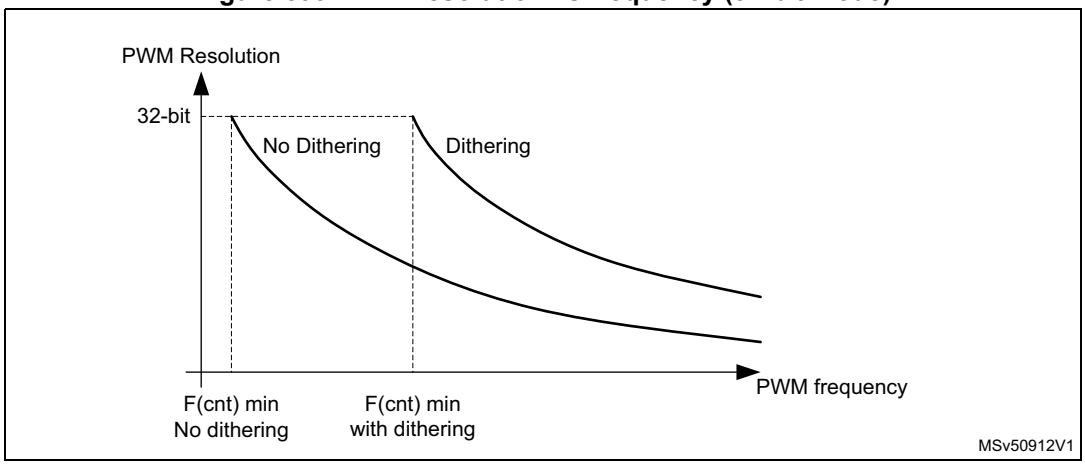
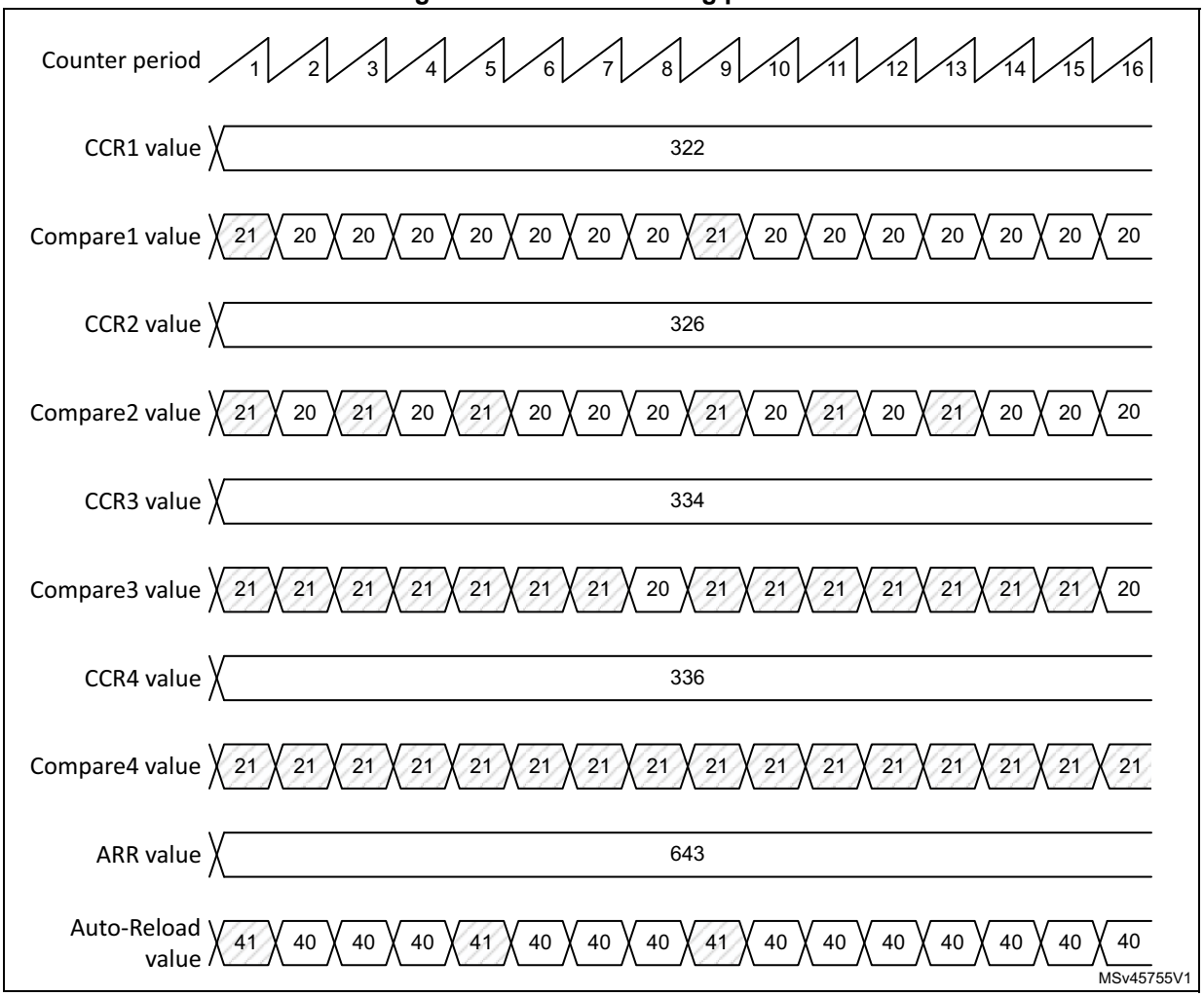


Figure 599. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 600*.

Figure 600. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 485*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 485. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 601*.

Figure 601. Dithering effect on duty cycle in center-aligned PWM mode

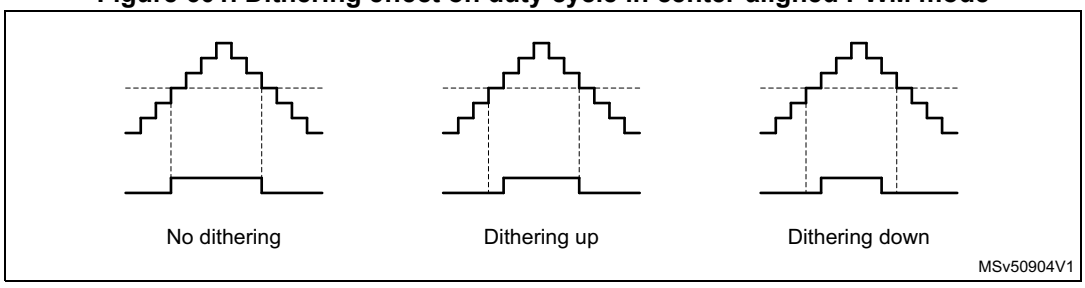


Table 486 shows how the dithering pattern is added in center-aligned PWM mode.

Table 486. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

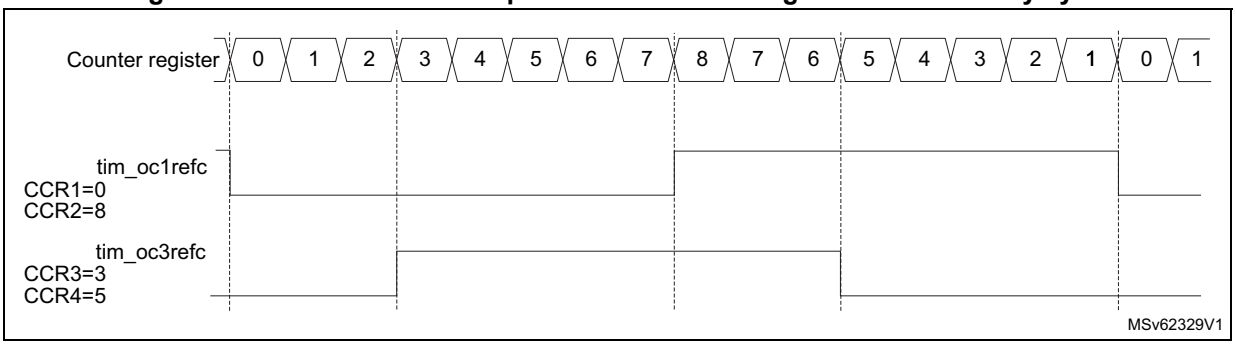
42.4.13 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle, for instance:

- tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2.
- tim_oc3refc is controlled by TIM_CCR3 and TIM_CCR4.

Figure 602 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

Figure 602. Generation of two phase-shifted PWM signals with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 487*. The selection is done with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (e.g. Asymmetric PWM mode 1 and 2), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 and 2, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 01110 for the asymmetric PWM mode 1 on oc1refc.

For the asymmetric PWM modes 7 to 10, it is mandatory to have opposite PWM mode for both channels (one being in PWM mode 1 and the other in PWM mode 2). Refer to *Table 487* for configuration.

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Table 487. Asymmetric PWM modes(1)

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	PWM Mode(2)
Asymmetric PWM mode 1	01110	oc1ref (up-counting) oc2ref (down-counting)		oc3ref (up-counting) oc4ref (down-counting)		1
Asymmetric PWM mode 2	01111					2
Asymmetric PWM mode 7	10110	-	-	oc3refc = 1 (up-counting) oc3ref OR oc4ref (down-counting)	-	1(3)
Asymmetric PWM mode 8	10111	-	-	oc3refc = 0 (up-counting) oc3ref AND oc4ref (down-counting)	-	2(4)
Asymmetric PWM mode 9	11000	-	-	oc3ref OR oc4ref (up-counting) oc3refc = 1 (down-counting)	-	1(3)
Asymmetric PWM mode 10	11001	-	-	oc3ref AND oc4ref (up-counting) oc3refc = 0 (down-counting)	-	2(4)

1. Asymmetric PWM mode 7 to 10 are only available on OC3M.

2. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value. PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the two channels used for the asymmetric mode.
3. oc4ref must be programmed in PWM mode 1.
4. oc4ref must be programmed in PWM mode 2.

42.4.14 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs,

- tim_oc1refc (or tim_oc2refc) is controlled by TIM_CCR1 and TIM_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIM_CCR3 and TIM_CCR4

Table 488 summarizes different combinations of PWM combined mode:

Table 488. Combined PWM mode overview

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	oc3refc	oc4refc	PWM mode(1)
Combined PWM mode 1	01100	oc1ref OR oc2ref		oc3ref OR oc4ref		1
Combined PWM mode 2	01101	oc1ref AND oc2ref		oc3ref AND oc4ref		2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value). PWM mode 2: channel active when the counter is above the TIM_CCRx value.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 01100 (combined PWM mode 1) or 01101 (combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

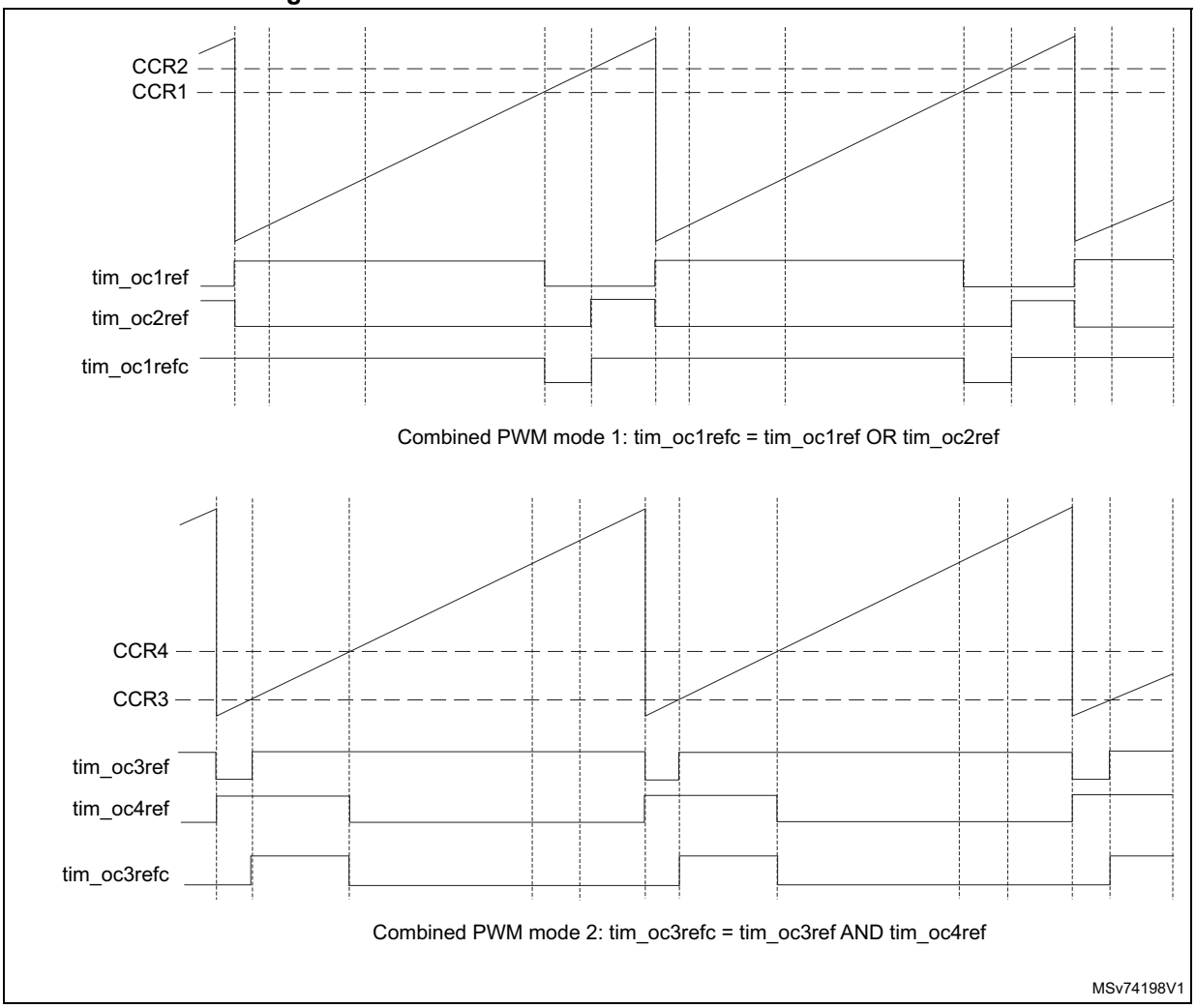
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in PWM mode 1 and the other in PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

Figure 603 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 1.
- Channel 2 is configured in PWM mode 2.
- Channel 3 is configured in combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 603. Combined PWM mode on channels 1 and 3

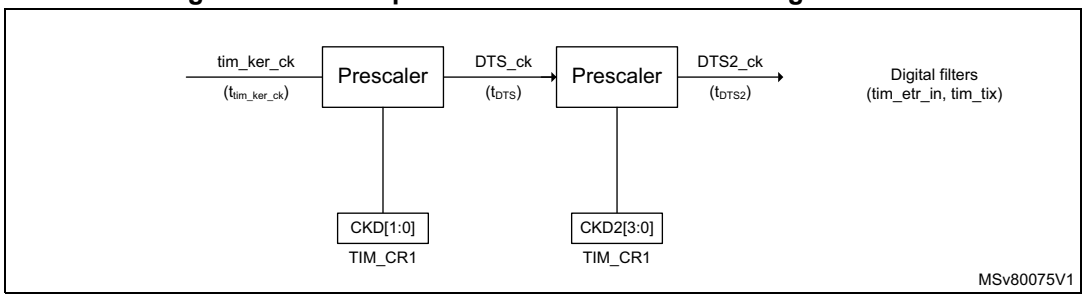


42.4.15 Clock dividers for dead-time generators and digital filters

The digital filters (for tim_etr_in, tim_tix) use a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the DTS_ck clock period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 604. Clock prescalers for dead-time and digital filters



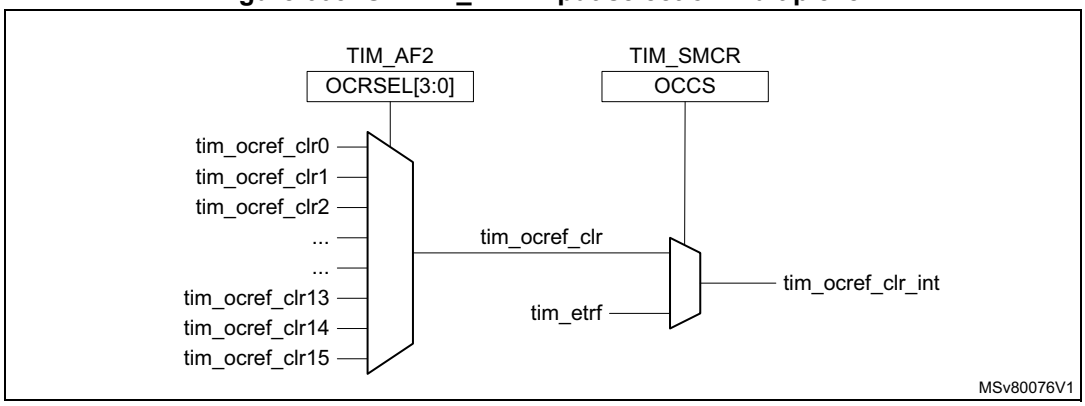
42.4.16 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in output compare and PWM modes. It must not be used in forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 42.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIM_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[15:0] inputs, using the OCRSEL[3:0] bitfield in the TIM_AF2 register, as shown in *Figure 605*.

Figure 605. OCREF_CLR input selection multiplexer



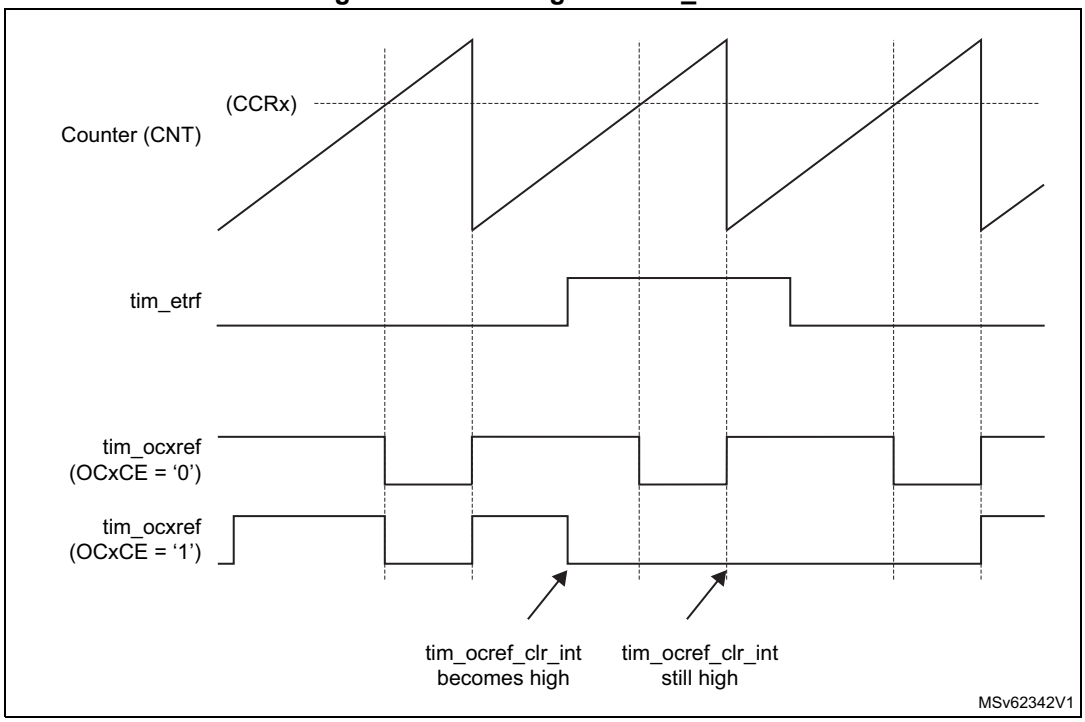
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIM_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 606 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 606. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to enable the OCxCE bit for the channel. The OCxCE bit of the complementary channel (for exemple: OC2CE = X for oc2ref in *Figure 607*) is ignored. *Figure 607* to *Figure 609* describe the output waveforms for specific use cases when the ocref_clr_int signal is asserted.

Figure 607. Asymmetric PWM mode 1

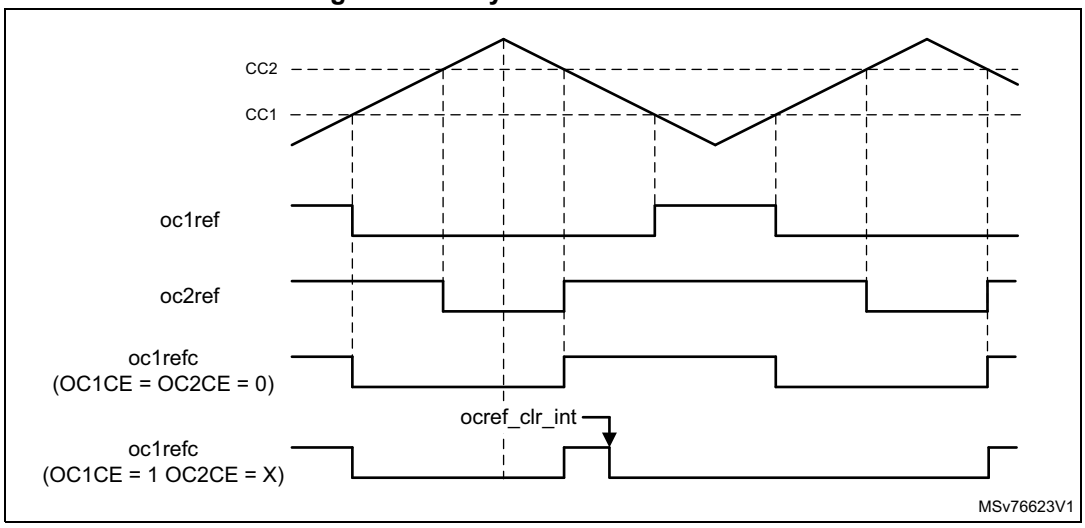


Figure 608. Asymmetric PWM mode 2

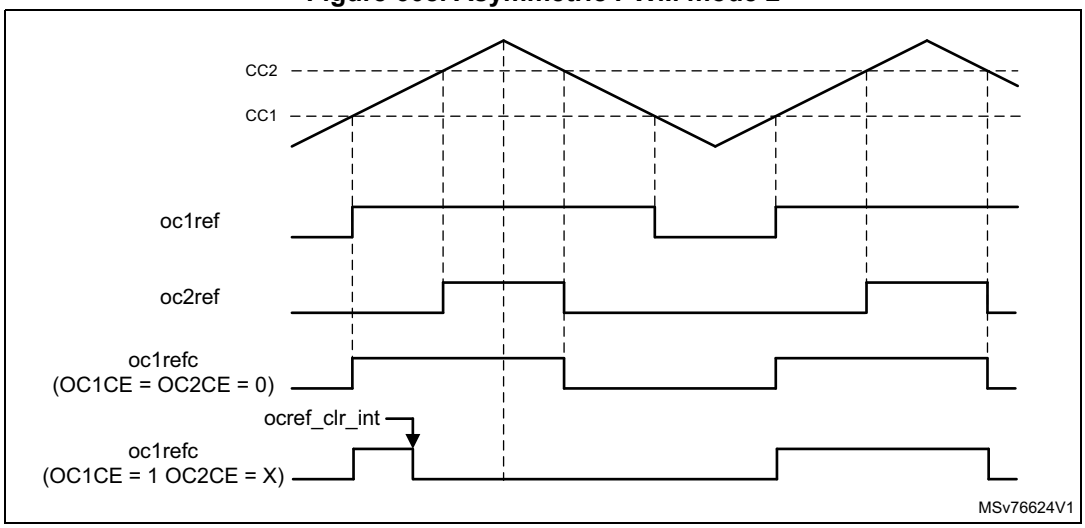
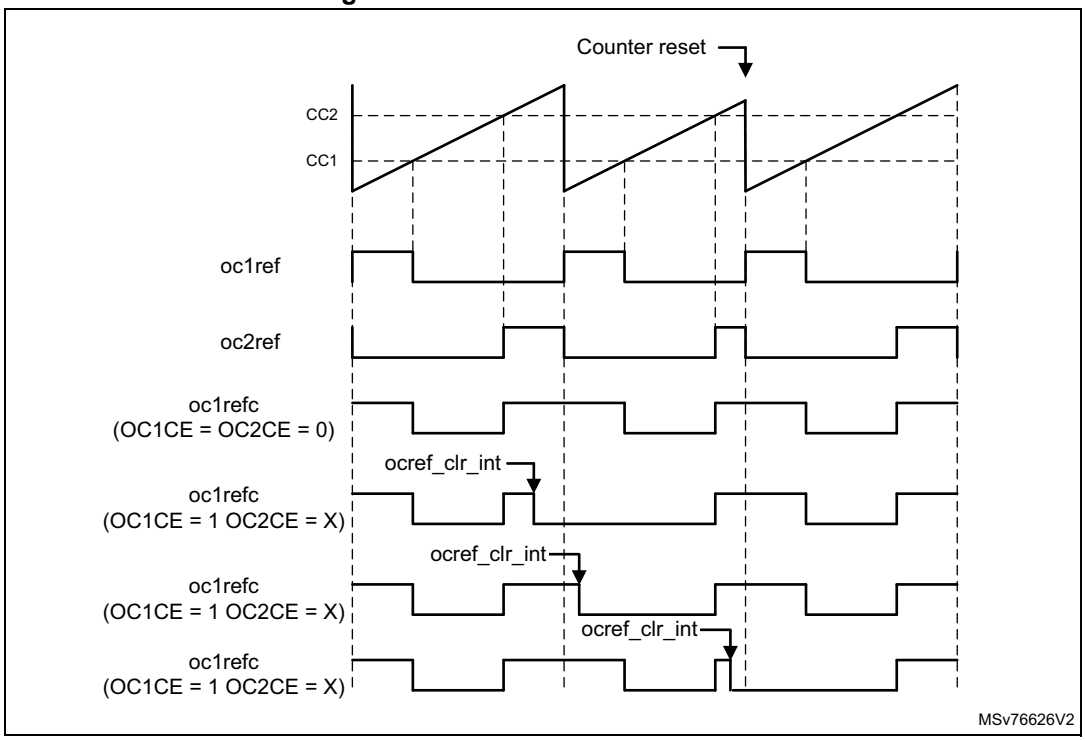


Figure 609. Combined PWM mode 1



Conditions of use

This feature supports cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used in the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One-pulse mode with PWM active on match (OPM = 1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E)
- Combined PWM mode 1 or 3 in center-aligned mode (CMS[1:0] = 1, 2, 3 and OCxM[4:0] = 0x0C)

42.4.17 One-pulse mode

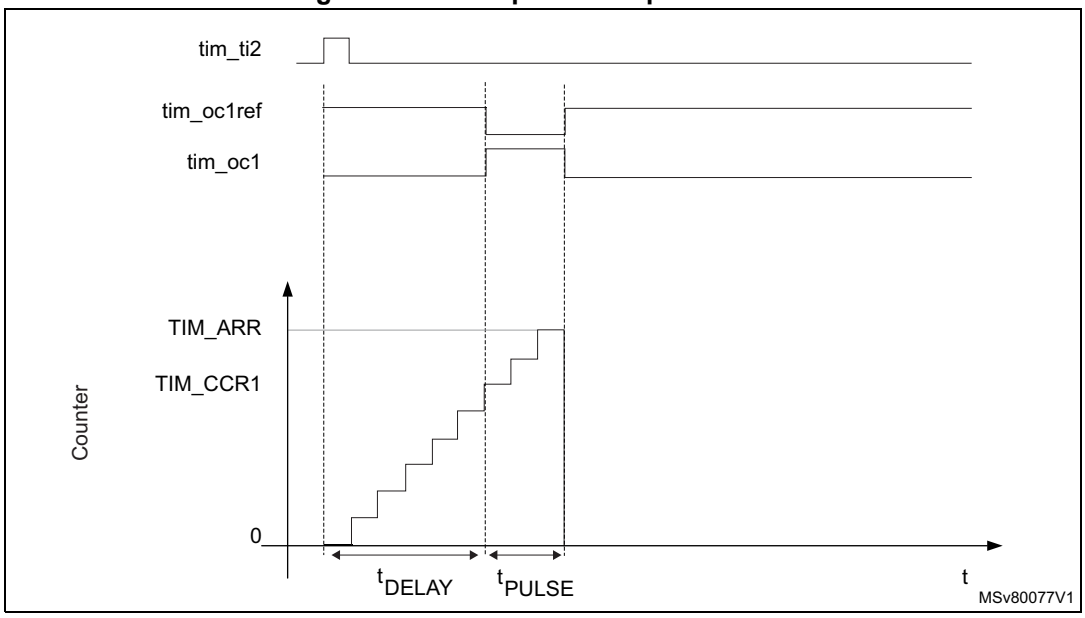
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 610. Example of one-pulse mode



For example if the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 00111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIM_CR1 register must be low.

Since only one pulse (single mode) is needed, a one must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the repetitive mode is selected.

Particular case: tim_ocx fast enable:

In one-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

42.4.18 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in *Section 42.4.17*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in slave mode, with the bits SMS[4:0] = 01000 (combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for retriggerable OPM mode 1 or 2.

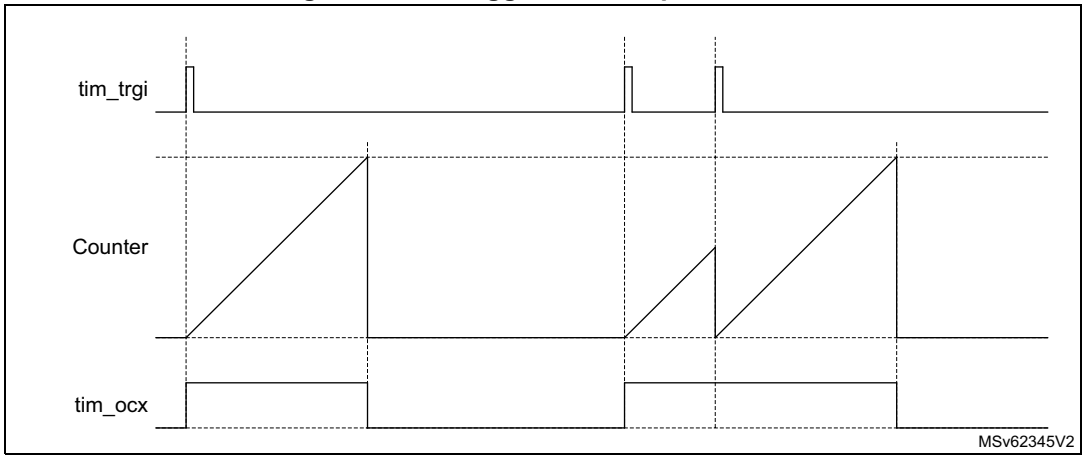
If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[4:0] and SMS[4:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 611. Retriggerable one-pulse mode



42.4.19 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 612*.

Figure 612. Pulse generator circuitry

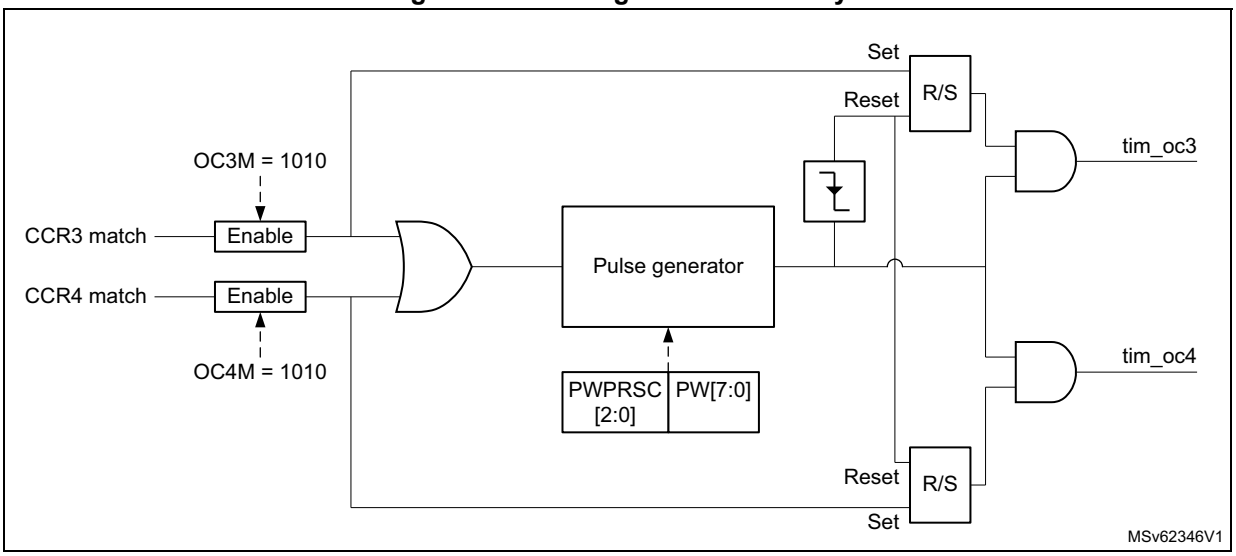
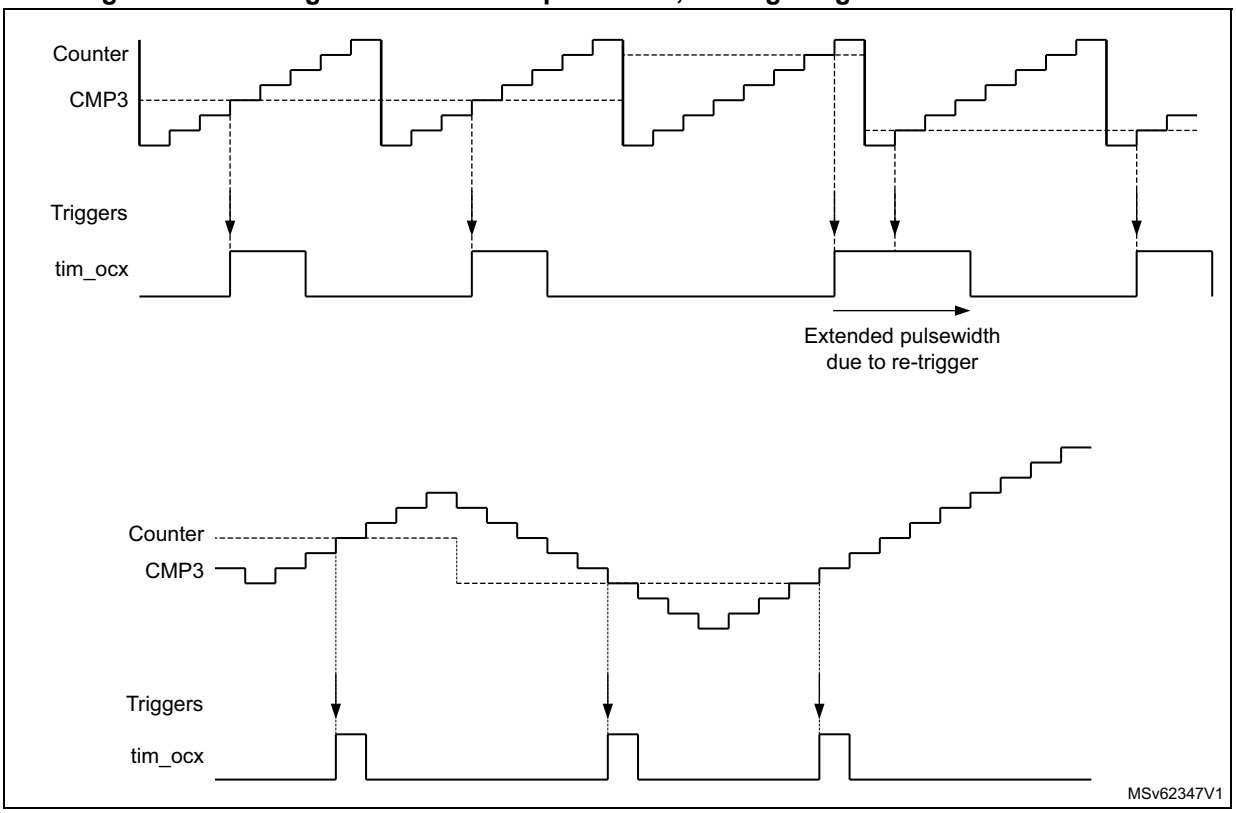


Figure 613 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 613. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIM_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

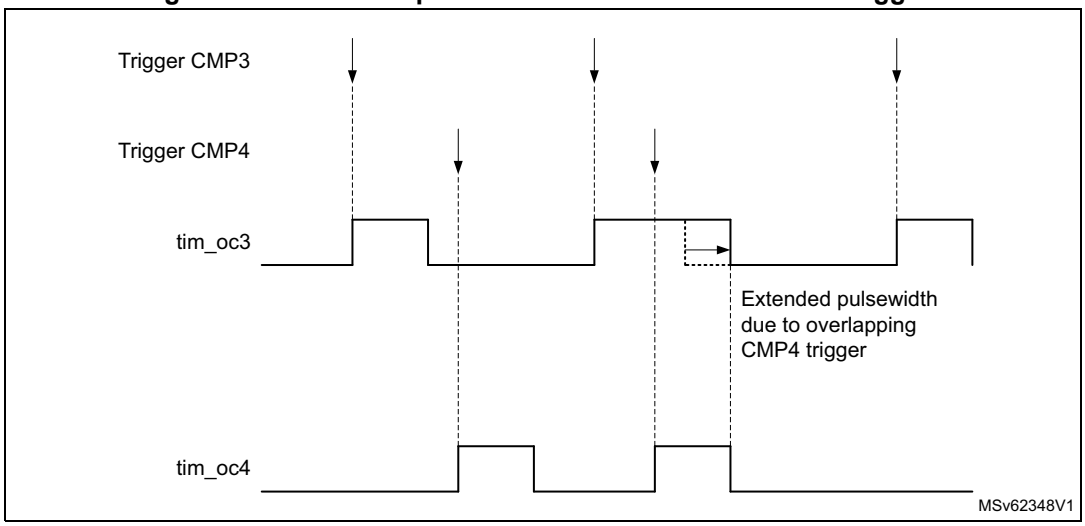
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

Gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 614).

Figure 614. Extended pulse width in case of concurrent triggers



42.4.20 Encoder interface mode

Quadrature encoder

To select encoder interface mode write SMS = 0001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 489*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and external clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 489. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 615 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIM_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, counter is enabled).

Figure 615. Example of counter operation in encoder interface mode

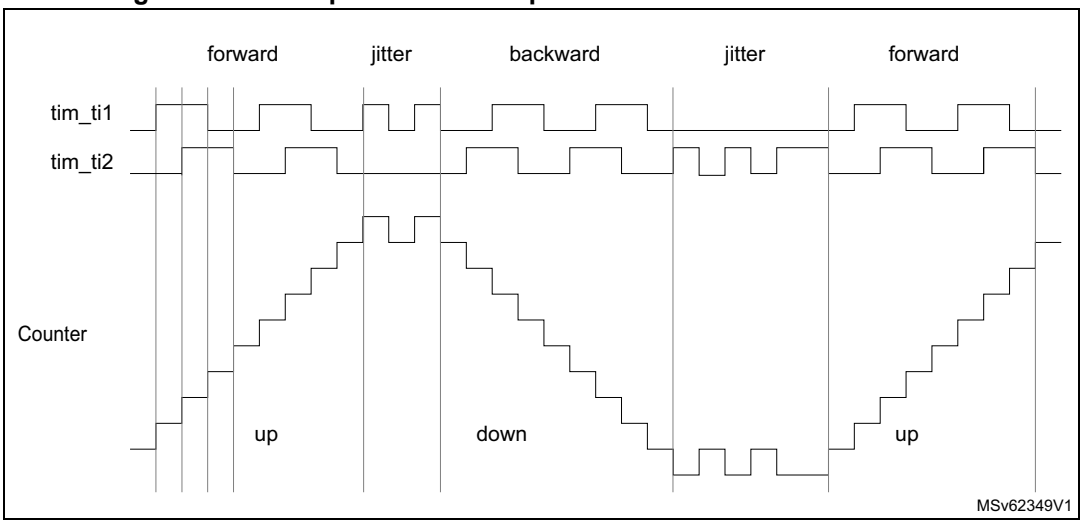


Figure 616 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 616. Example of encoder interface mode with tim_ti1fp1 polarity inverted

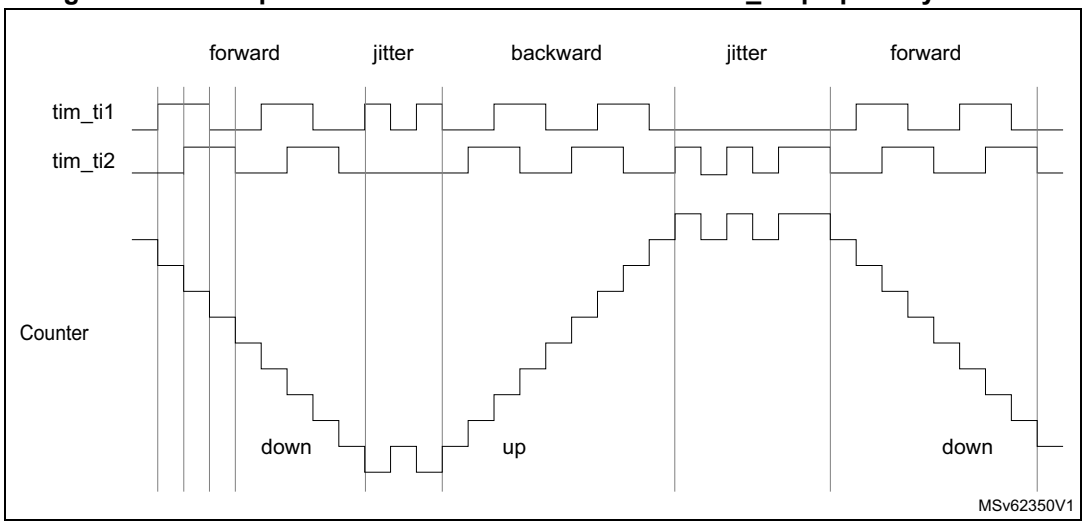
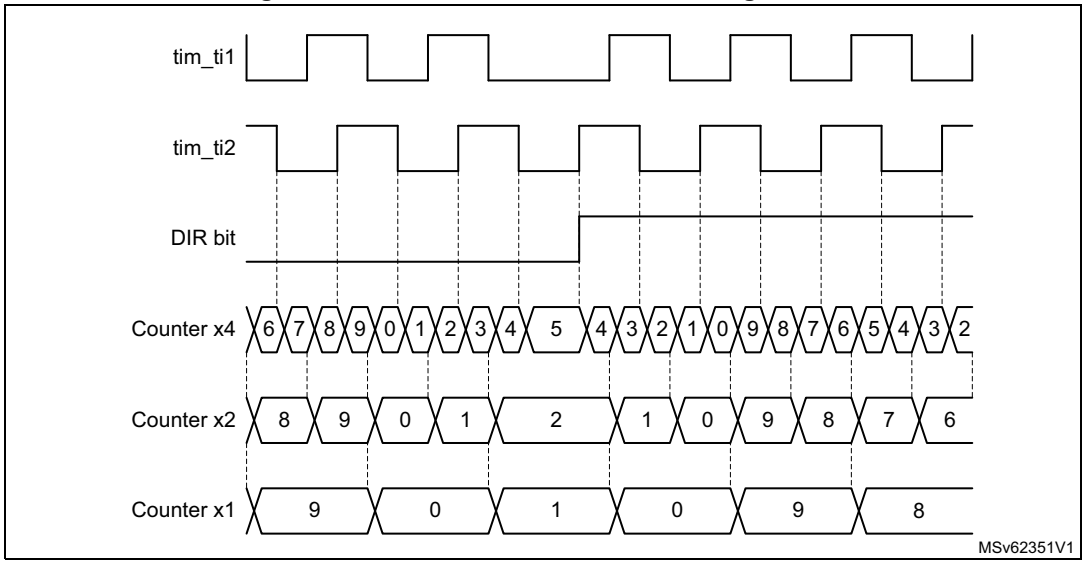


Figure 617 shows the timer counter value during a speed reversal, for various counting modes.

Figure 617. Quadrature encoder counting modes



The timer, when configured in encoder interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

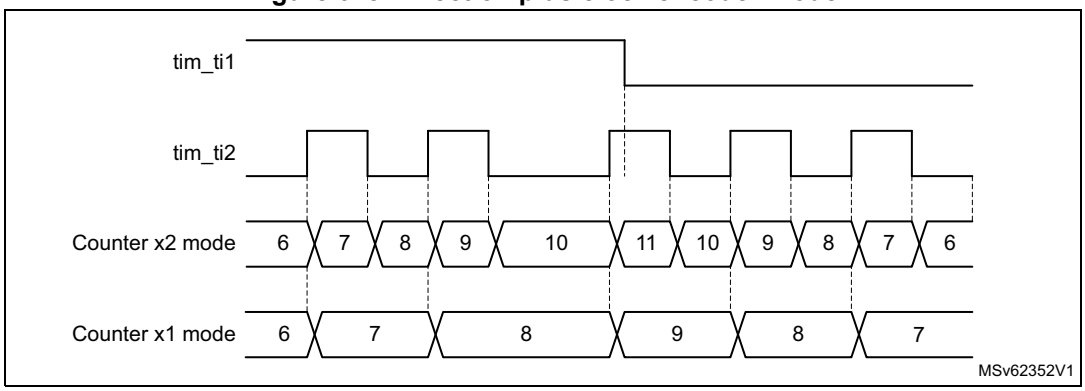
In the "clock plus direction" mode shown on *Figure 618*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 618. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 619*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 619*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 620*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 619*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 620*).

Figure 619. Directional clock encoder mode (CC1P = CC2P = 0)

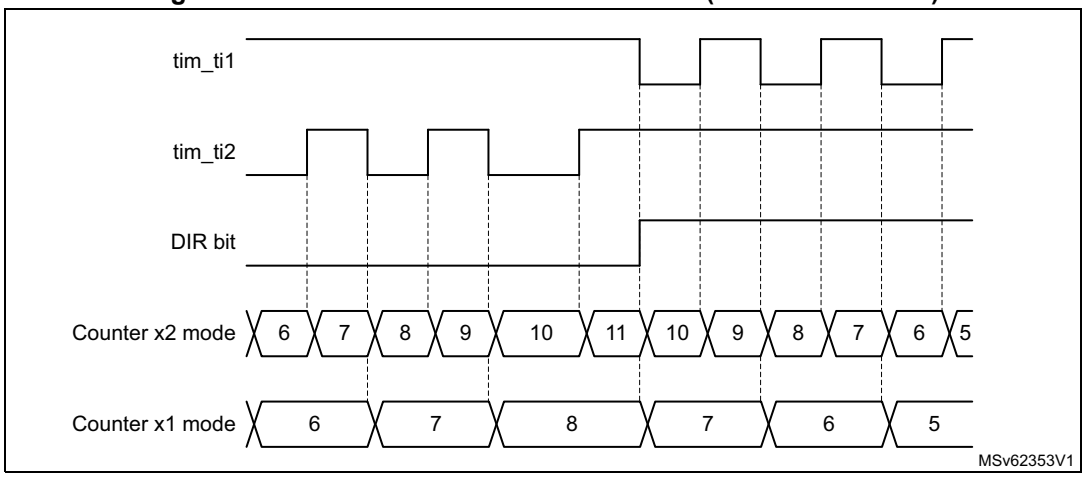


Figure 620. Directional clock encoder mode (CC1P = CC2P = 1)

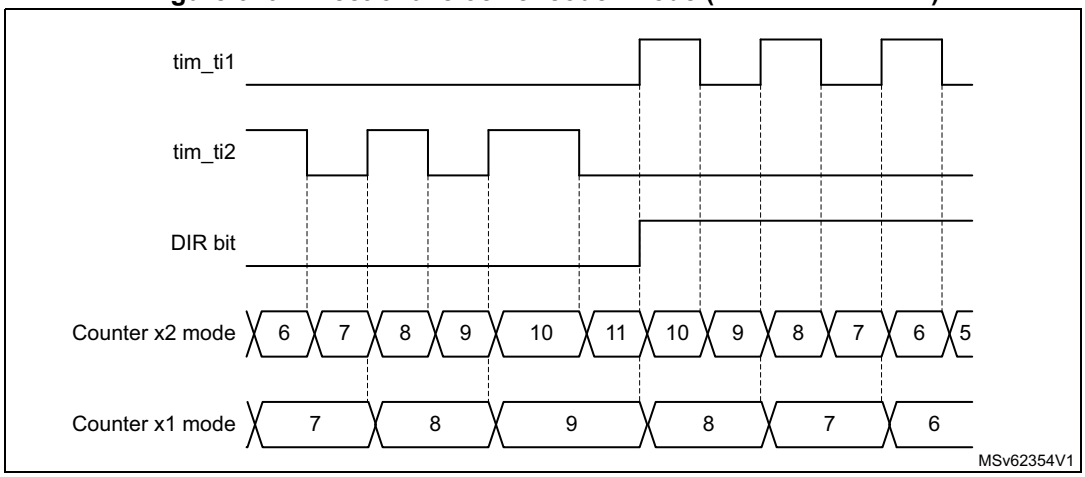


Table 490 details how the directional clock mode operates, for any input transition.

Table 490. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

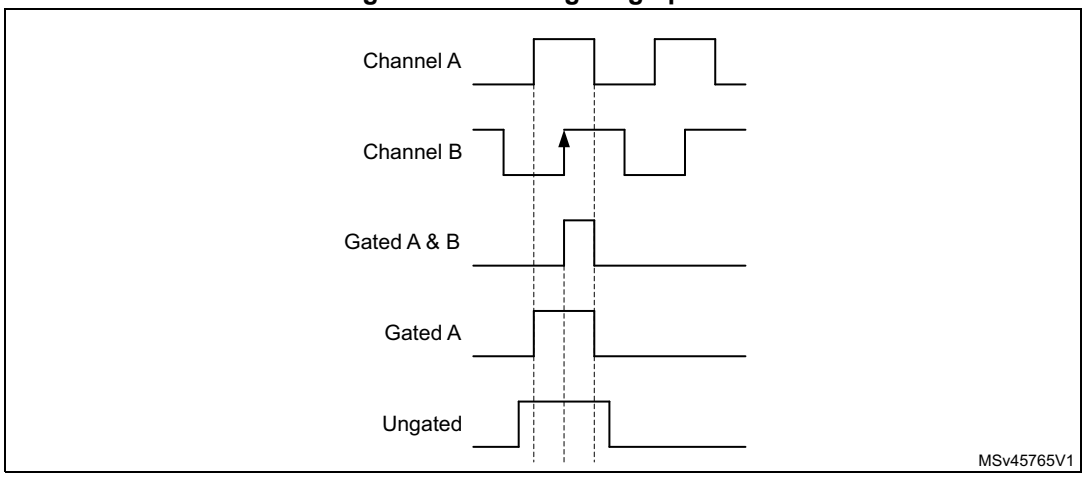
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIM_ECR register. The IE bit must be set only in encoder mode, when the SMS[4:0] bitfield has the following values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 621*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

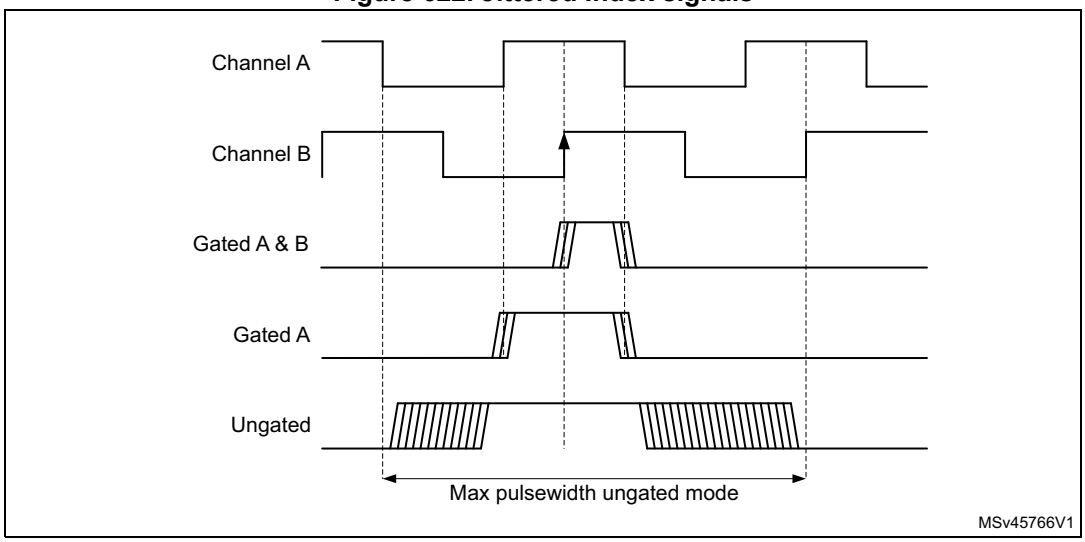
Figure 621. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 622*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 622. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIM_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIM_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 623* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 623. Index generation for IPOS[1:0] = 11

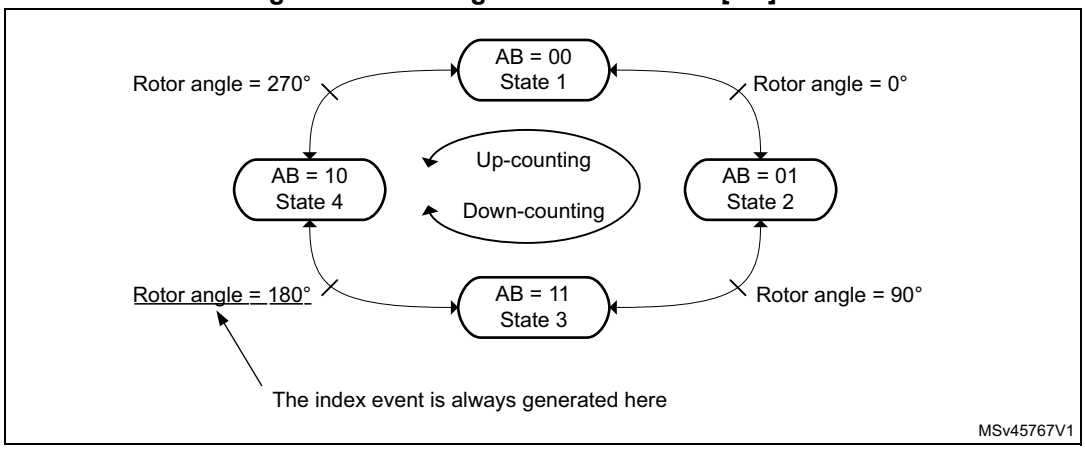


Figure 624 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIM_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 624. Counter reading with index gated on channel A (IPOS[1:0] = 11)

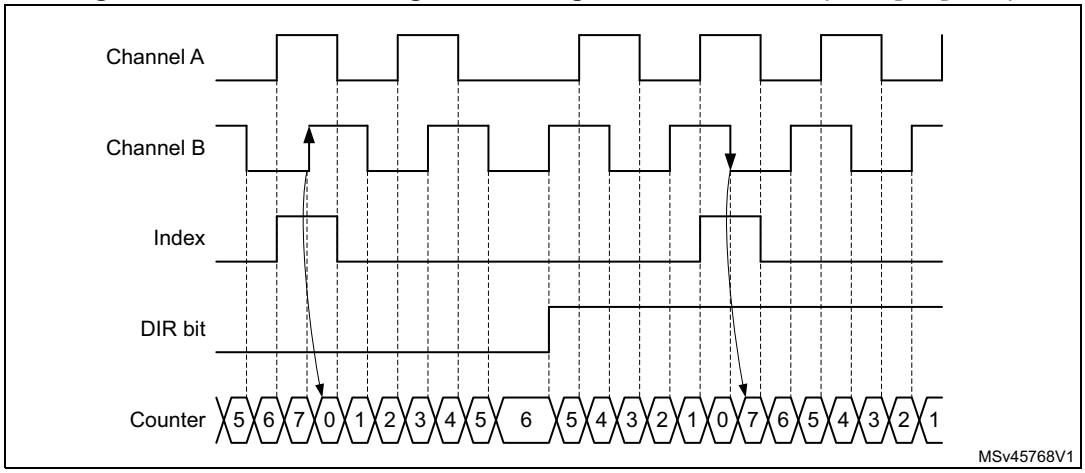


Figure 625 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 625. Counter reading with index ungated (IPOS[1:0] = 00)

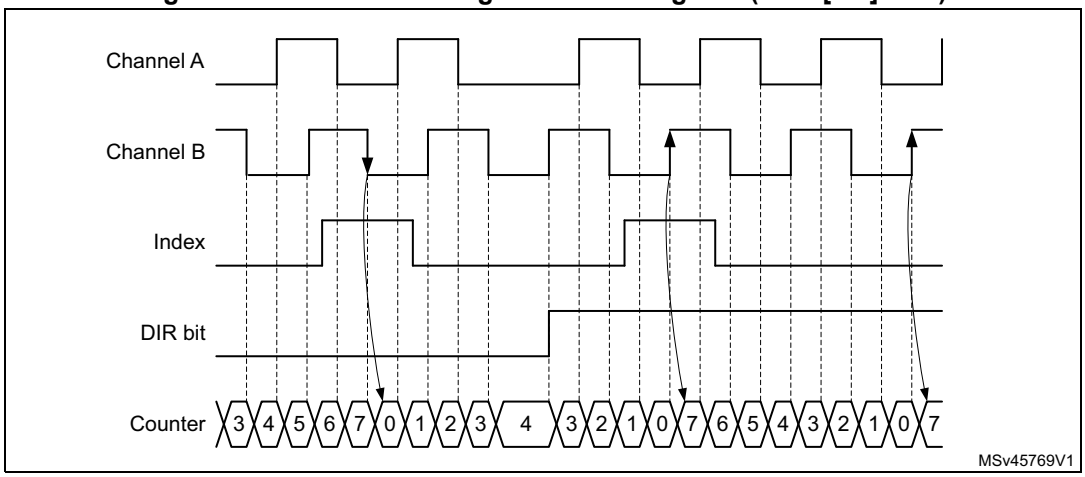


Figure 626 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 626. Counter reading with index gated on channel A and B

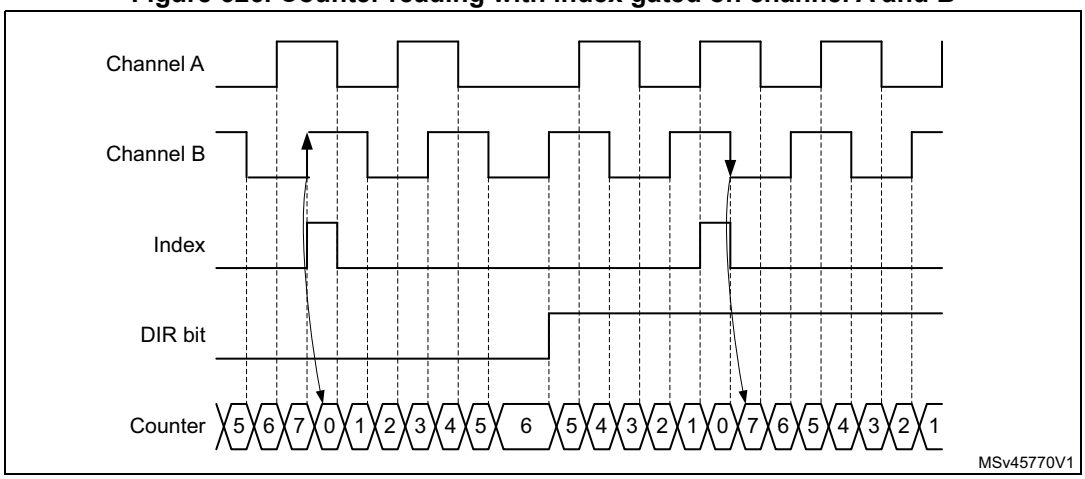


Figure 627 and *Figure 628* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 627. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

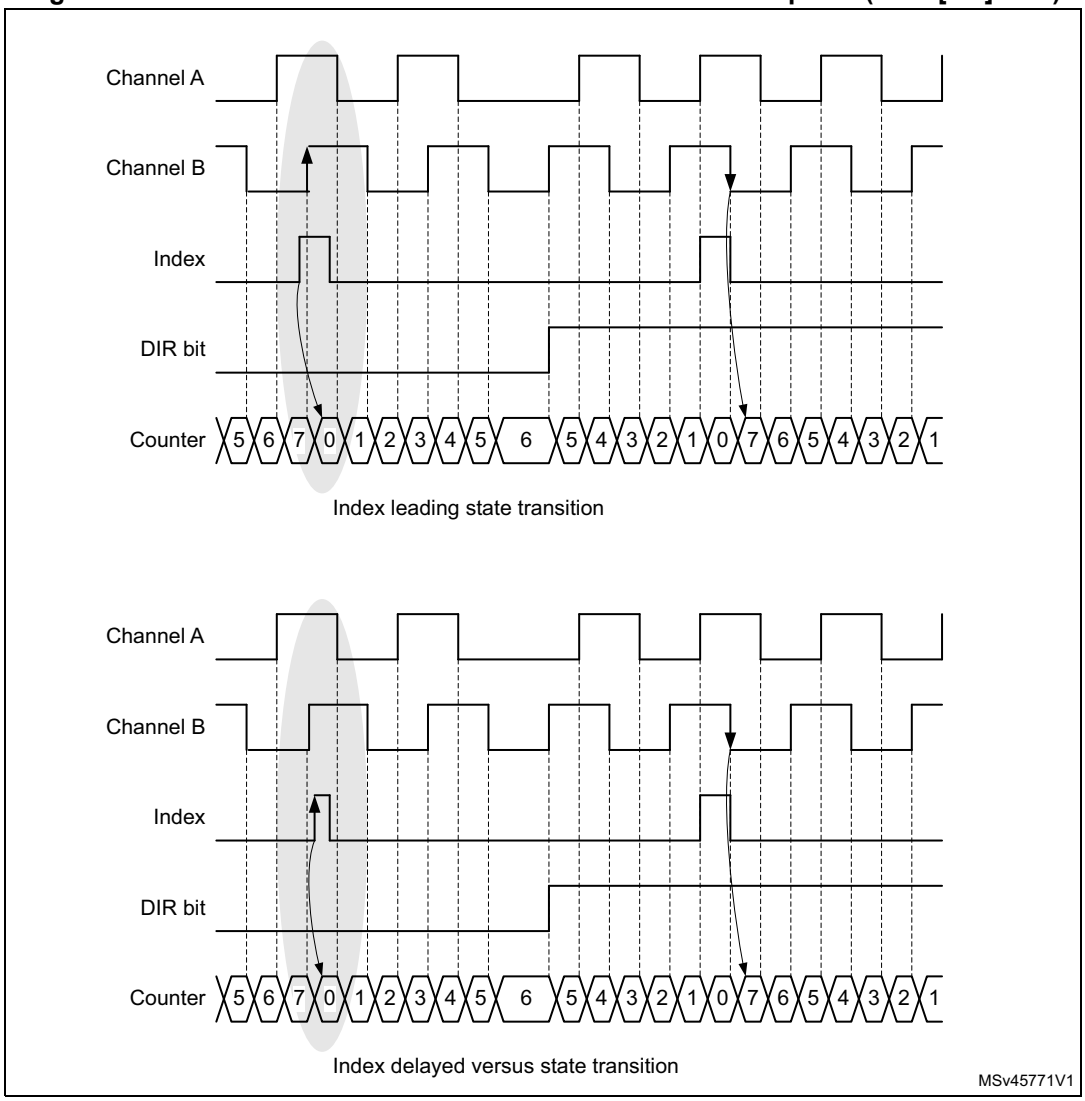


Figure 628. Counter reset Narrow index pulse (closer view, ARR = 0x07)

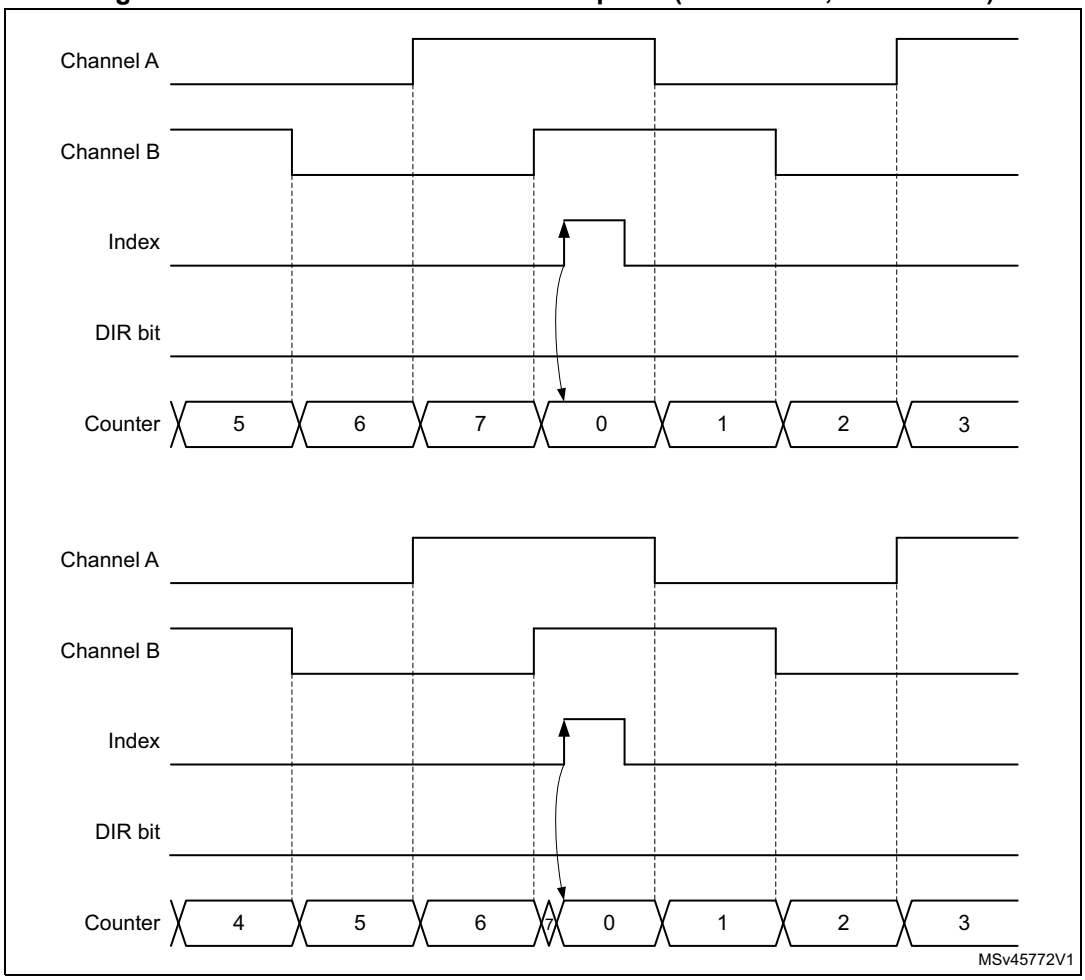
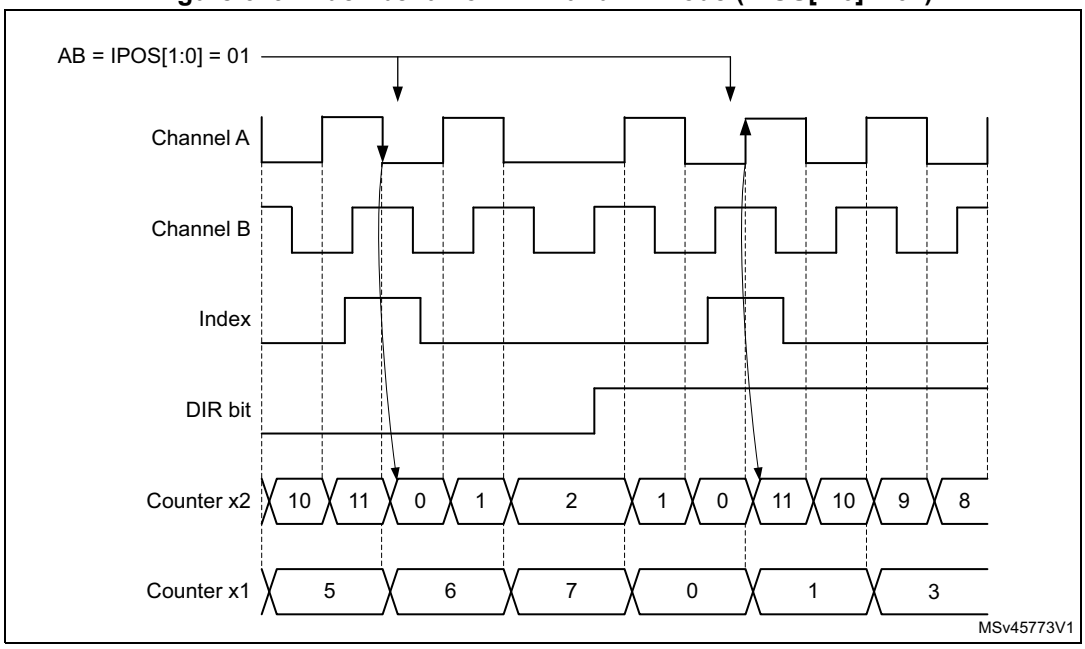


Figure 629 shows how the index is managed in x1 and x2 modes.

Figure 629. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

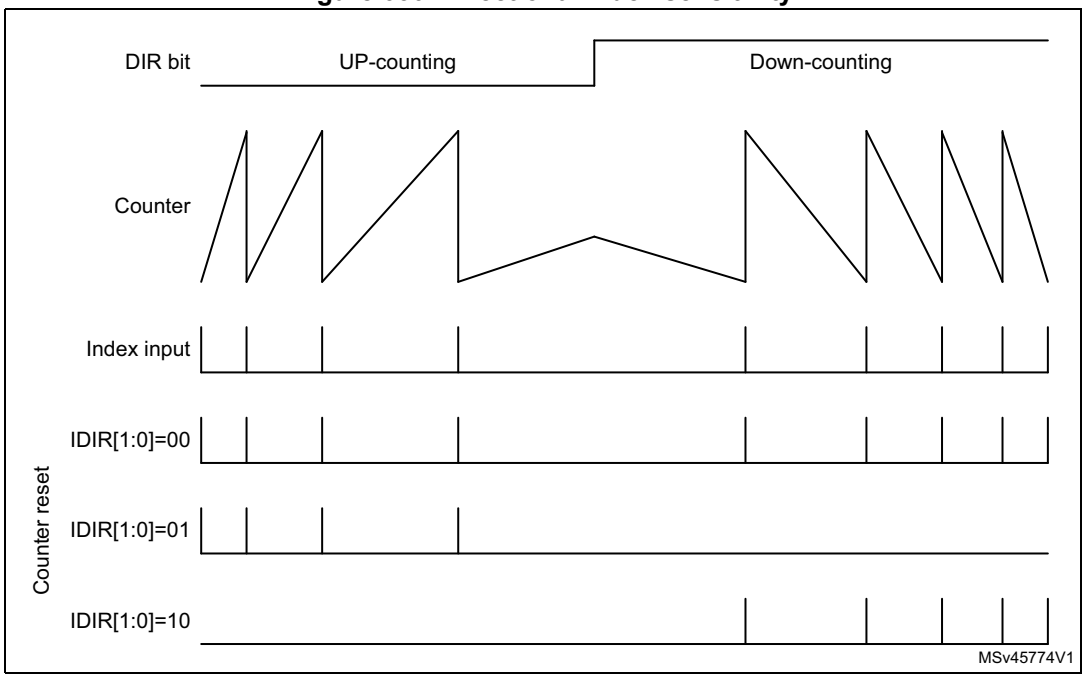


Directional index sensitivity

The IDIR[1:0] bitfield in the TIM_ECR register allows the index to be active only in a selected counting direction.

Figure 630 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

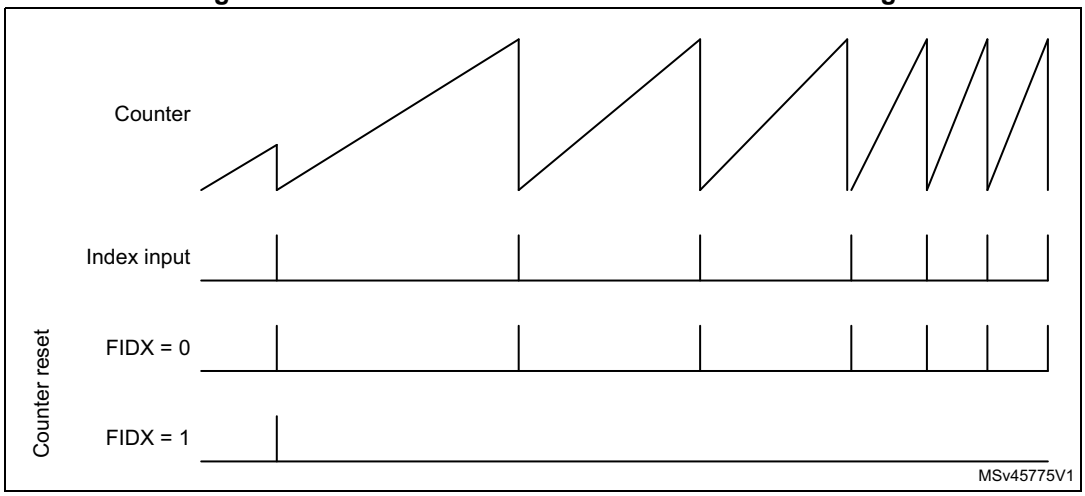
Figure 630. Directional index sensitivity



Special first index event management

The FIDX bit in the TIM_ECR register allows the index to be taken only once, as shown on *Figure 631*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 631. Counter reset as function of FIDX bit setting



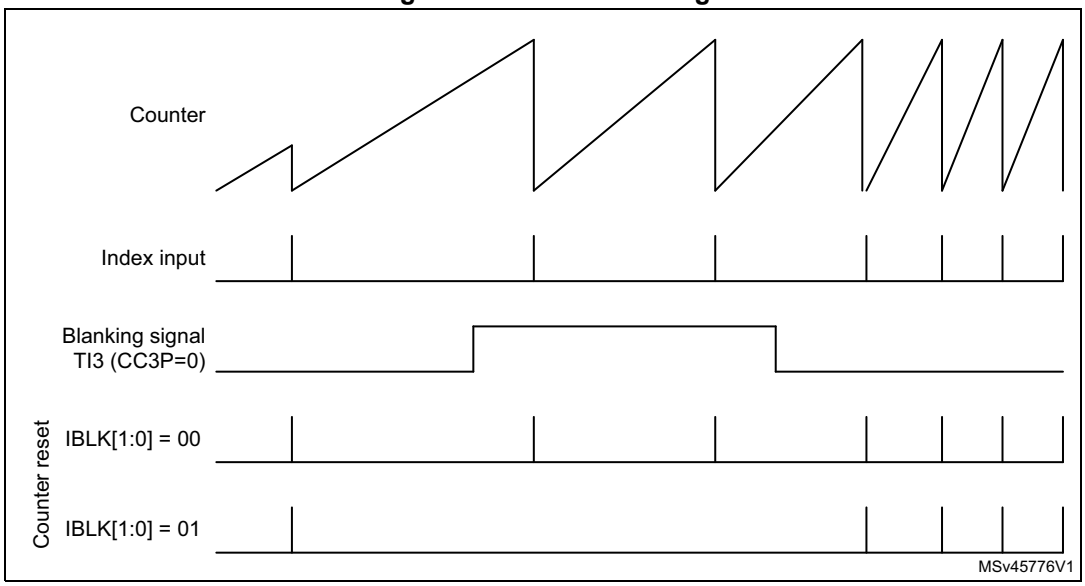
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on *Figure 632*.

This mode is enabled using the IBLK[1:0] bitfield in the TIM_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

Figure 632. Index blanking



Index management in nonquadrature mode

Figure 633 and *Figure 634* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[4:0] bitfield is equal to 01010, 01011, 01100, 01101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 633. Index behavior in clock + direction mode, IPOS[0] = 1

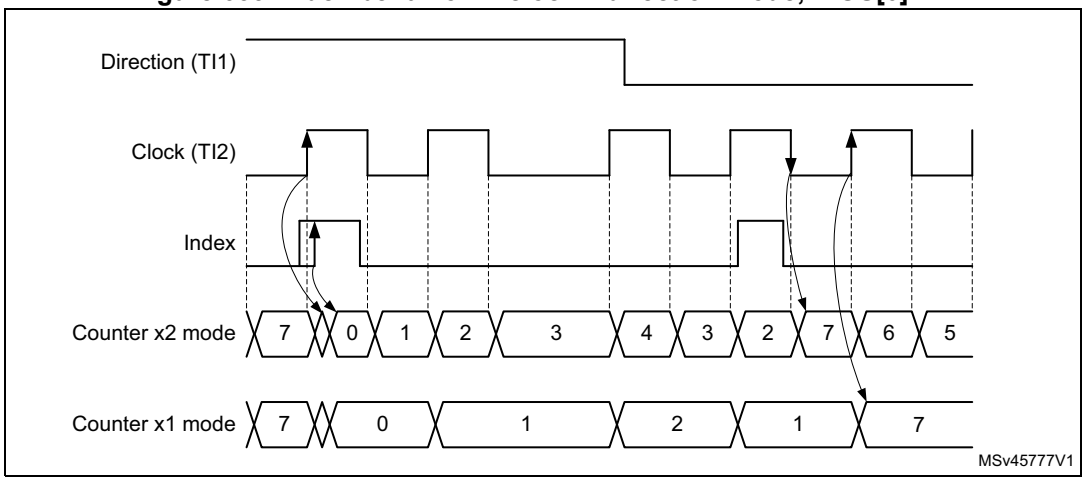
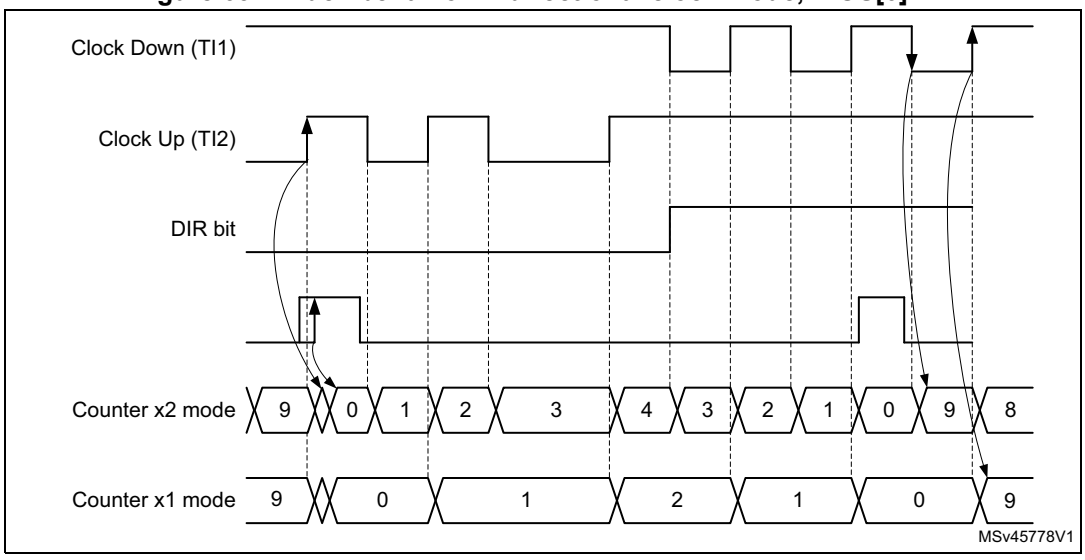


Figure 634. Index behavior in directional clock mode, IPOS[0] = 1

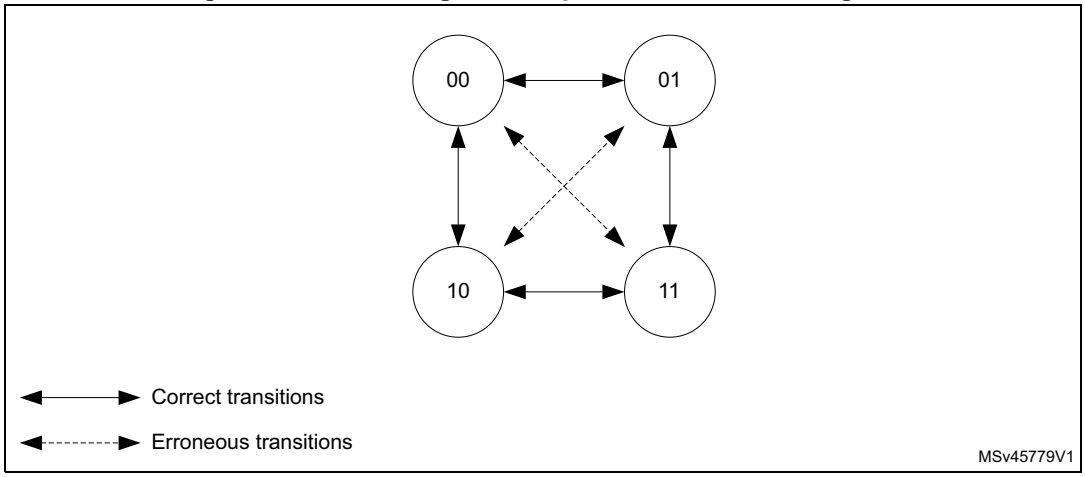


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 635*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status

register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 635. State diagram for quadrature encoded signals



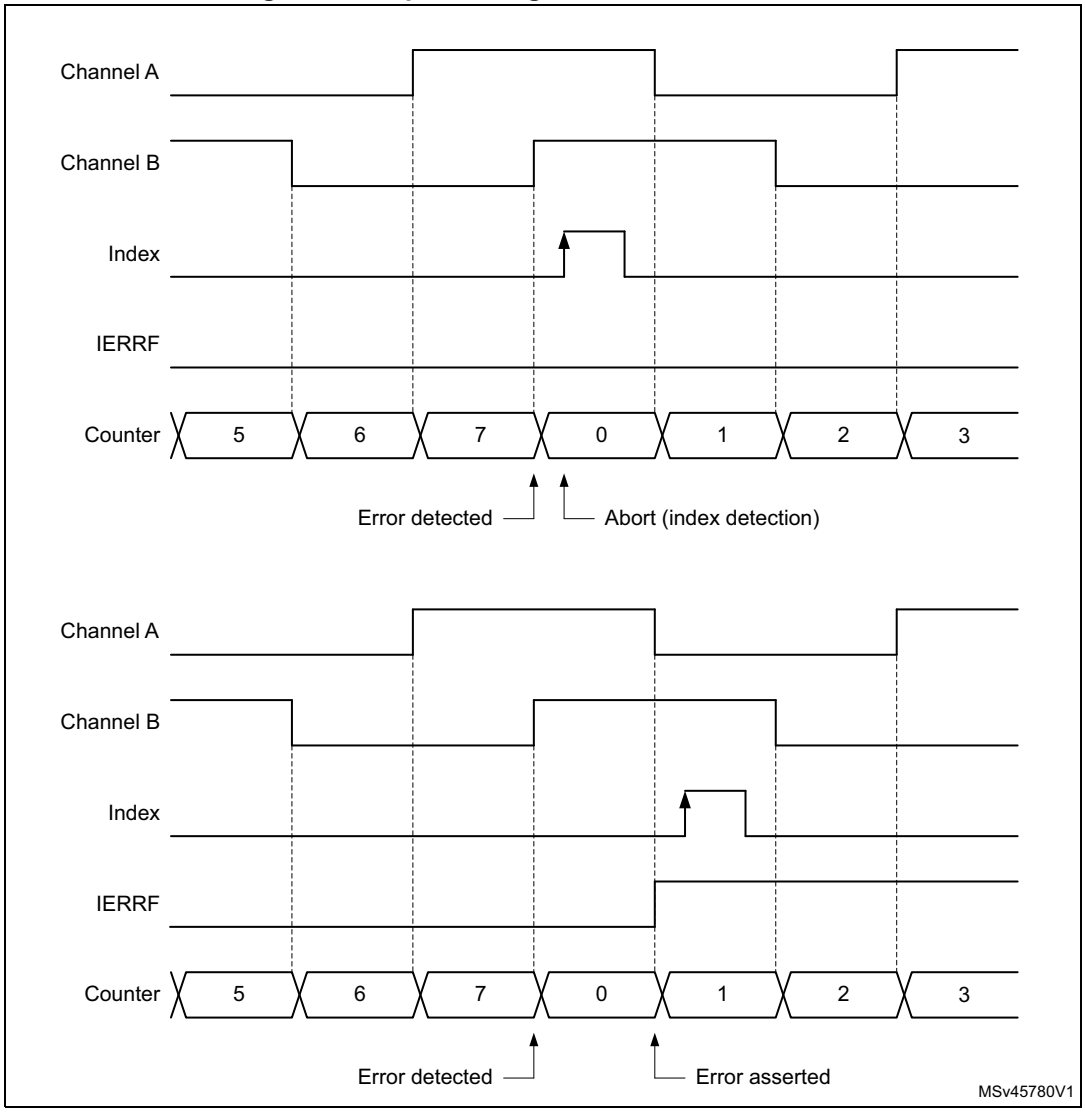
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIM_ARR to 0 or decremented from 0 to TIM_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIM_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIM_ARR = 3999 + 1 = 4000.

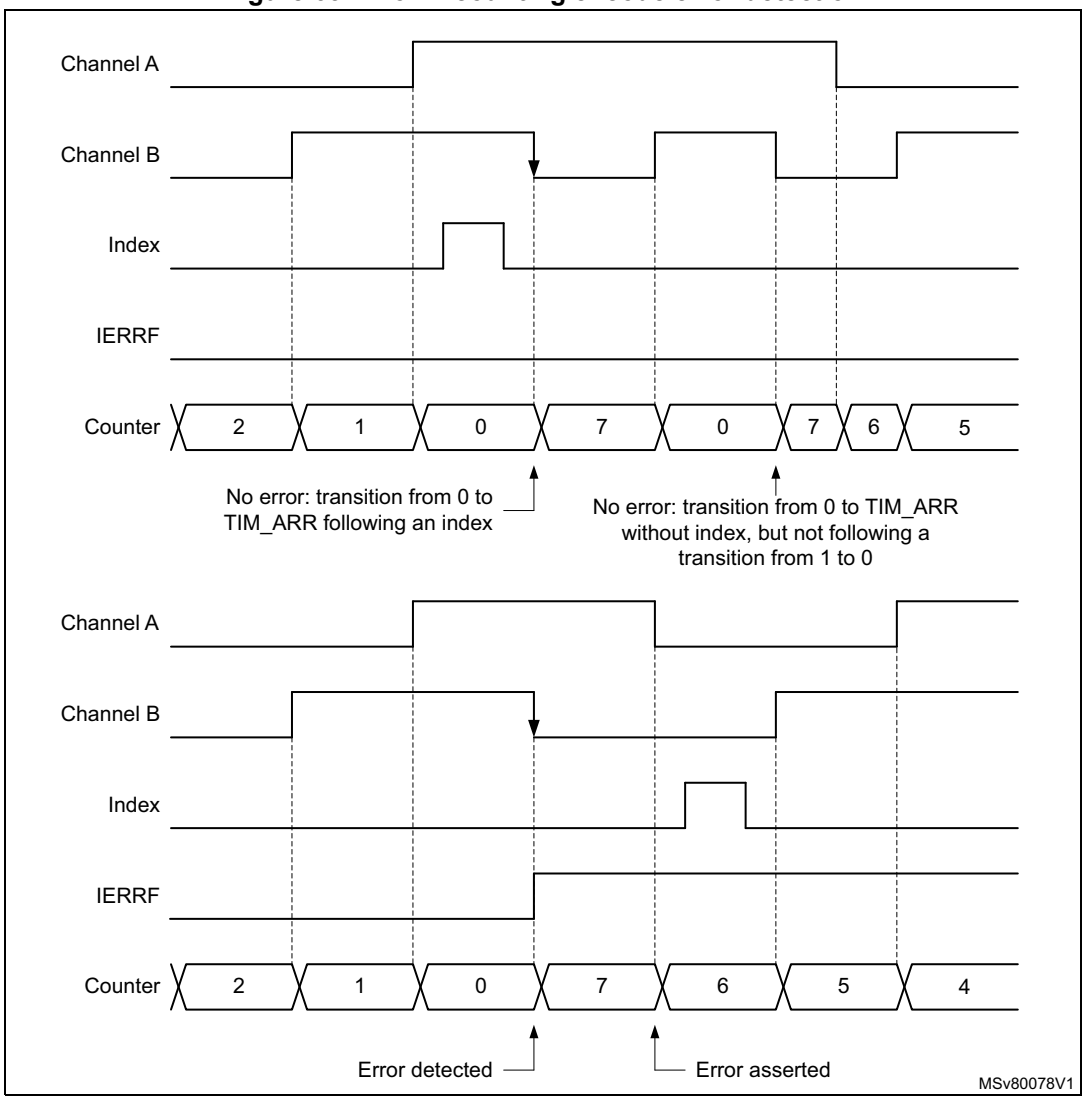
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 636*.

Figure 636. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 637*, to avoid any false error detection in case the encoder dithers between TIM_ARR and 0 immediately after the index detection.

Figure 637. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIM_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIM_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIM_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIM_DIER register.

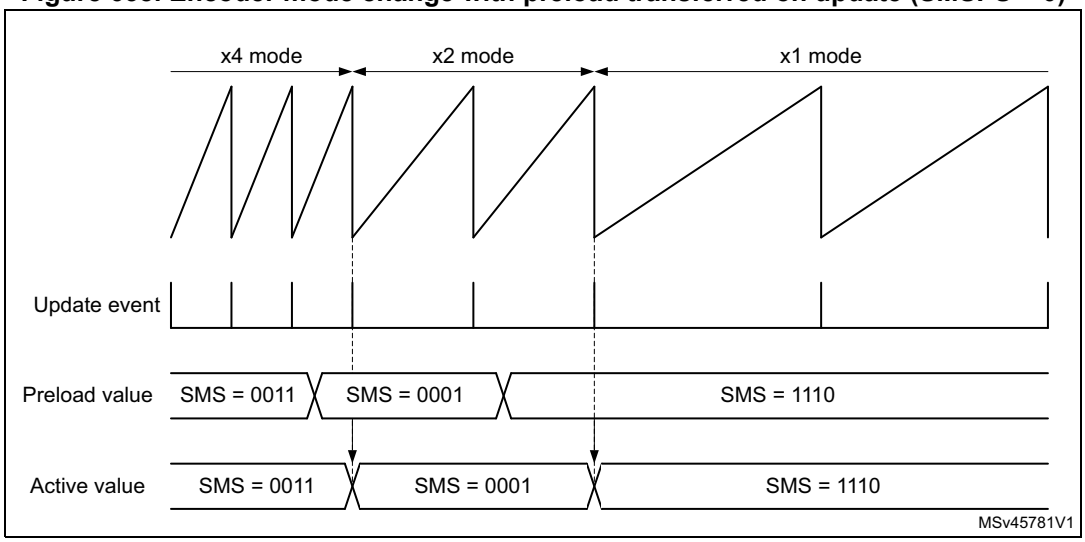
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 638*.

For this purpose, the SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value can be selected with the SMSPS bit in the TIM_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 638. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register. It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

42.4.21 Encoder with built-in debouncer

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 639* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there's no edge on input 2.

The *Figure 640* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between two positions.

These two modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 639. Encoder with built-in debouncer, x2 mode

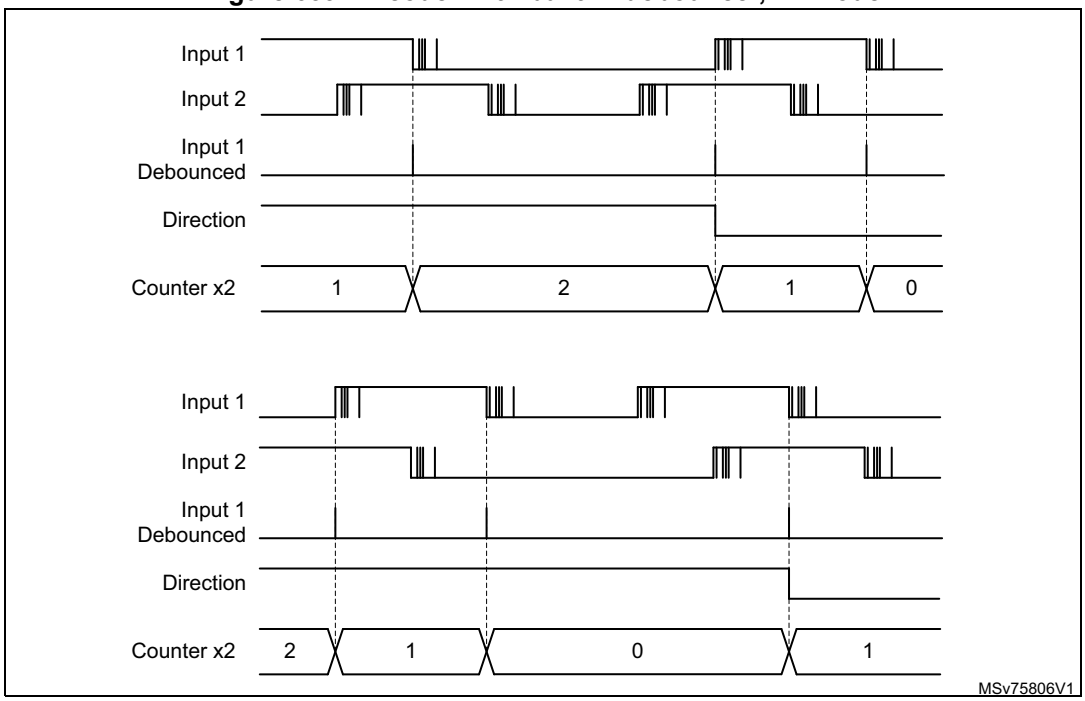
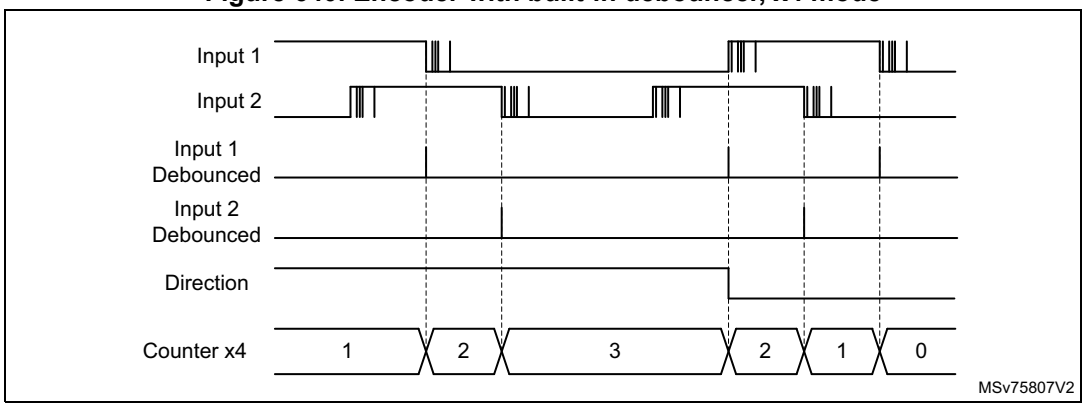


Figure 640. Encoder with built-in debouncer, x4 mode



42.4.22 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIM_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIM_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

42.4.23 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

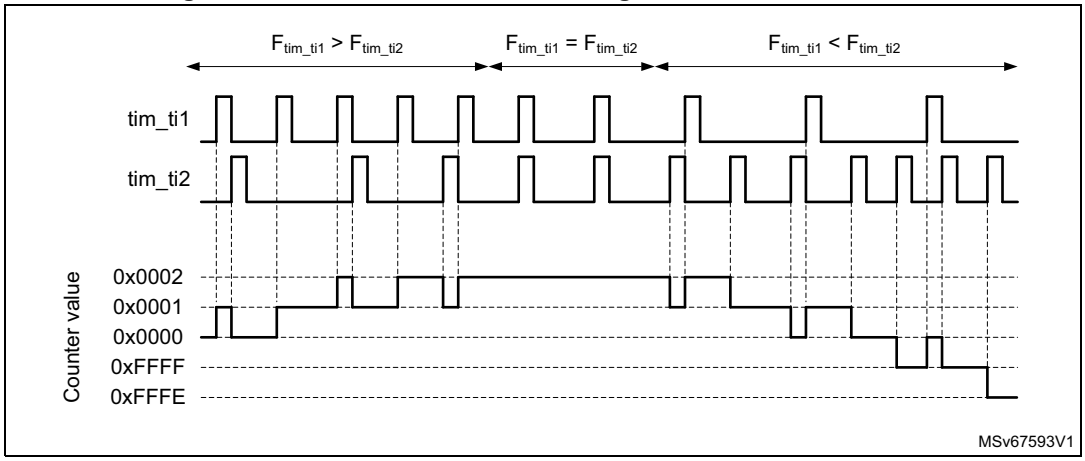
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 641*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value will change as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases.
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases.
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value will dither between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occurring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 641*).

Figure 641. Clock drift measure using directional clock mode



42.4.24 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

42.4.25 Timer input XOR function

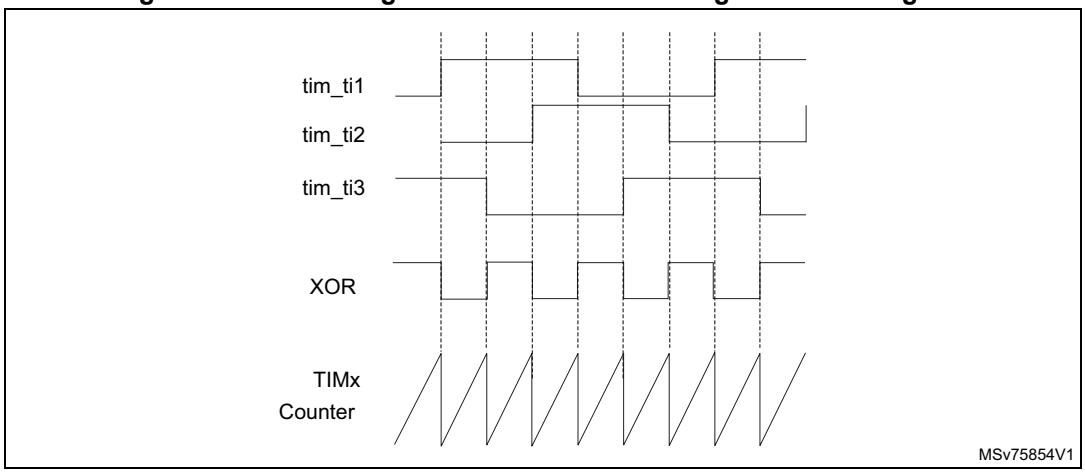
The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 41.3.34: Interfacing with Hall sensors*.

It is convenient to measure the interval between edges on two input signals, as per *Figure 642*.

Figure 642. Measuring time interval between edges on three signals



The XORPS control bit selects the position of the XOR gate, as shown on the *Figure 643* and *Figure 644* below, when the TI1S bit is set.

Figure 643. XOR gate connection with TI1S = 1 and XORPS = 0

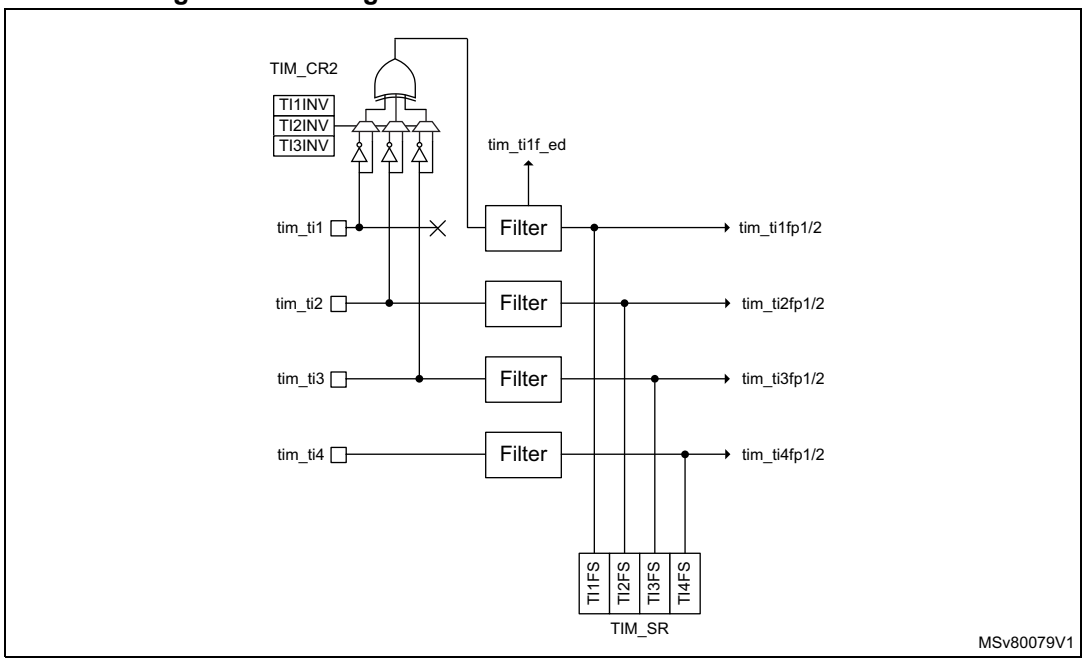
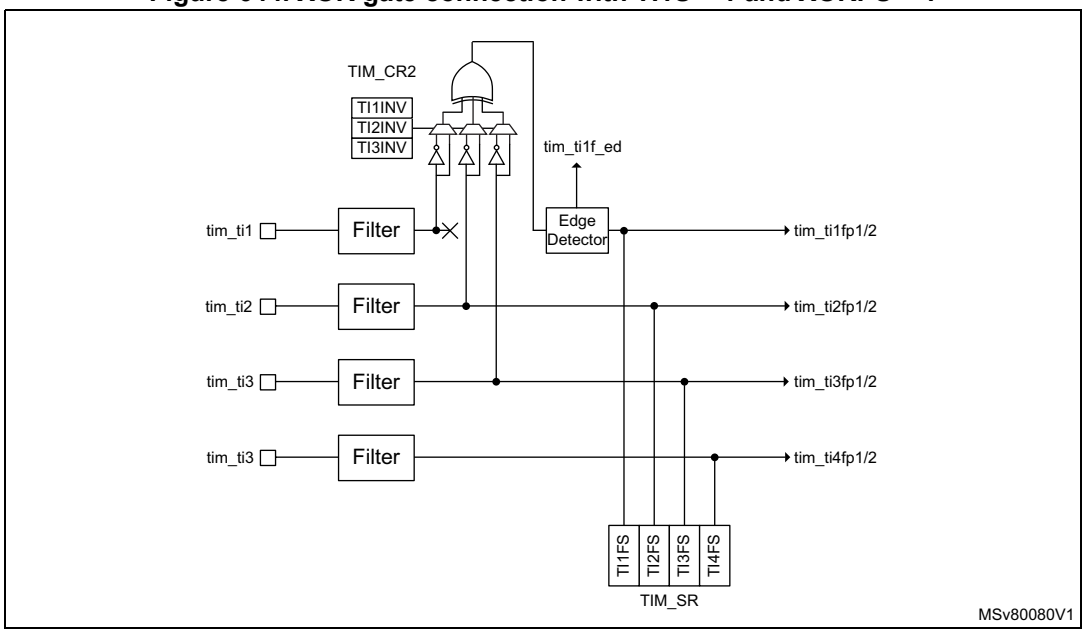


Figure 644. XOR gate connection with TI1S = 1 and XORPS = 1



Three control bits T1INV, T2INV and T3INV in the TIM_CR2 register can invert the tim_in1, tim_in2 and tim_in3 signals prior entering the XOR gate.

The status of the tim_in1, tim_in2, tim_in3 and tim_in4 signals can be software polled using the TI1FS, TI2FS, TI3FS and TI4FS bits in the TIM_SR register (tim_ti4 is not represented on figures above and do not enter into the XOR gate).

42.4.26 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: reset mode, gated mode, trigger mode, reset + trigger and gated + reset modes.

Slave mode: reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

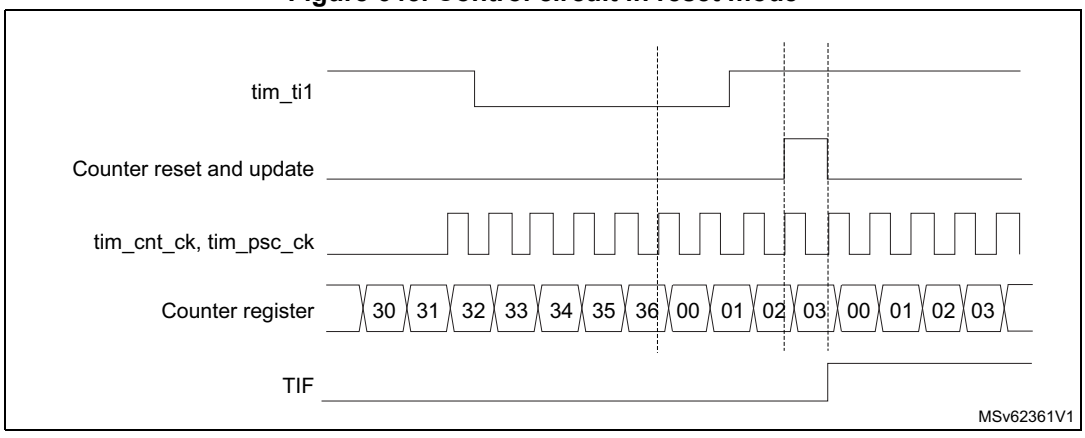
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 645. Control circuit in reset mode



Slave mode: gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The

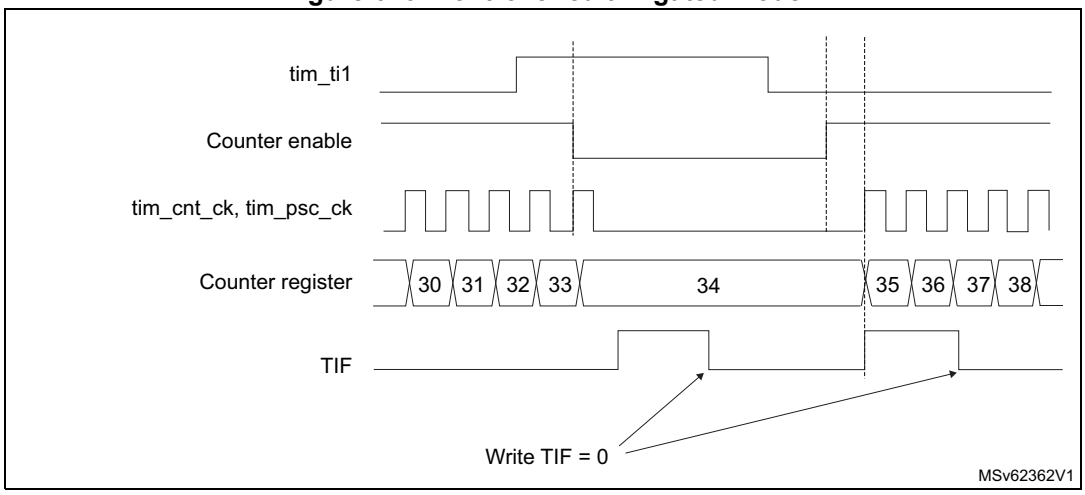
capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 646. Control circuit in gated mode



Note: The configuration "CCxP = CCxNP = 1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: trigger mode

The counter can start in response to an event on a selected input.

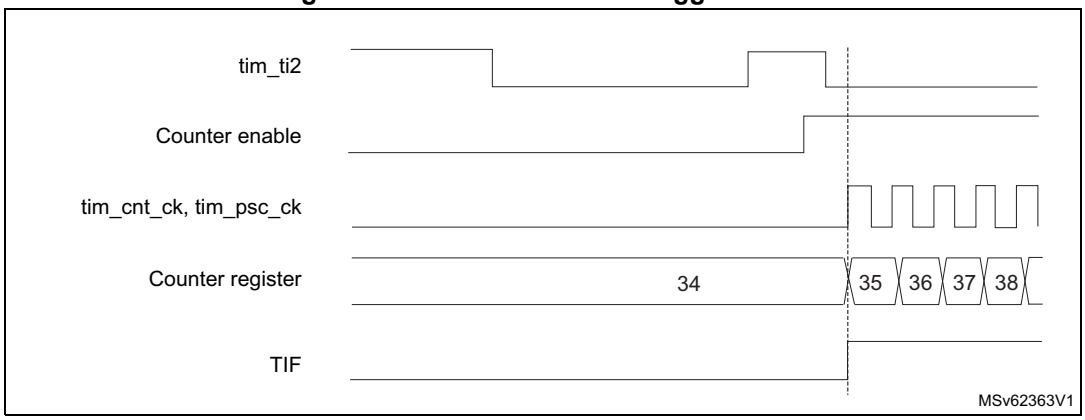
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 647. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIM_SMCR register.

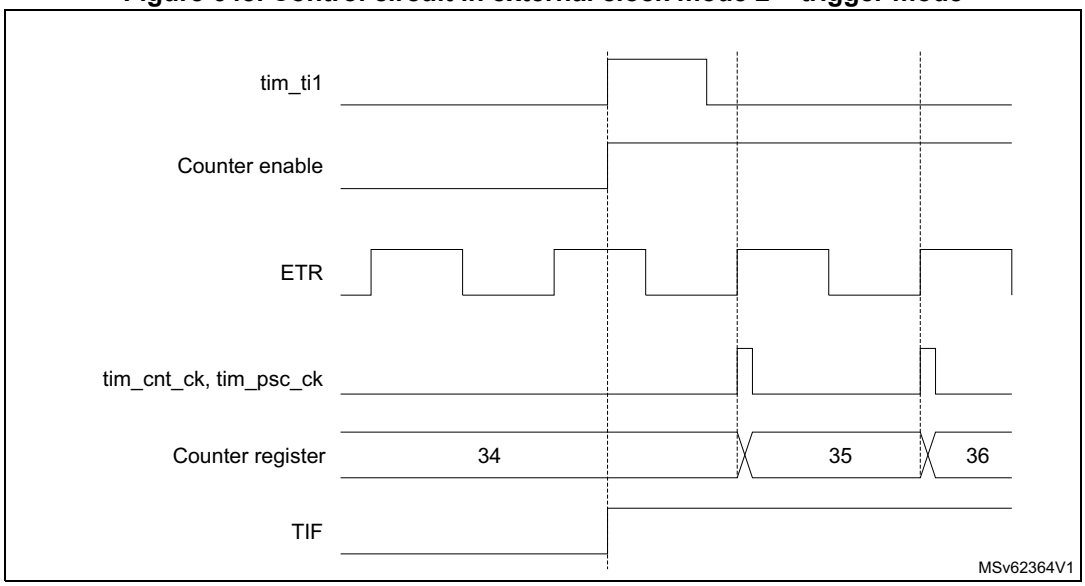
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIM_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIM_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 648. Control circuit in external clock mode 2 + trigger mode



42.4.27 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

Figure 649 and *Figure 650* show examples of master/slave timer connections.

Figure 649. Master/Slave timer example

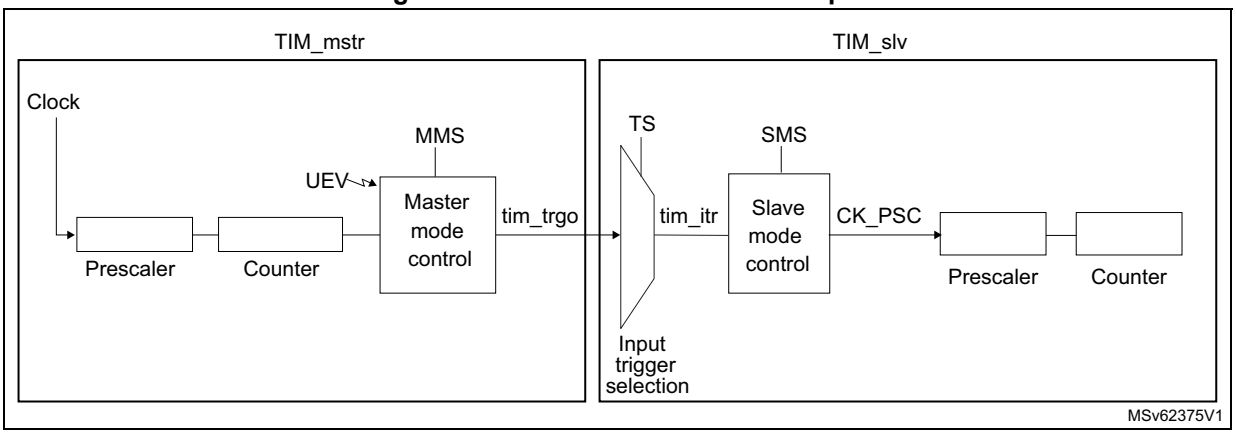
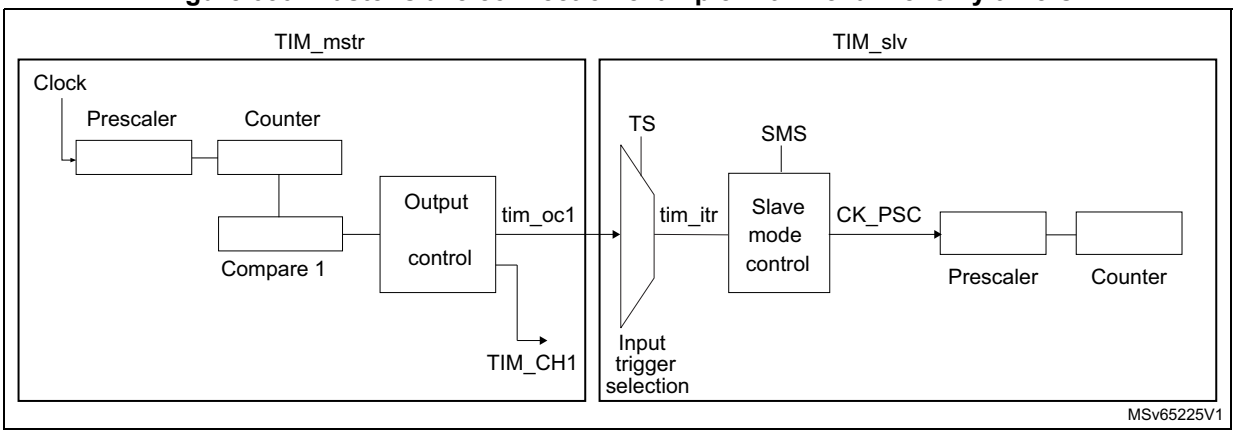


Figure 650. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 650) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals). The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger. For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 649*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge

of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).

4. Finally both timers must be enabled by setting their respective CEN bits (TIM_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

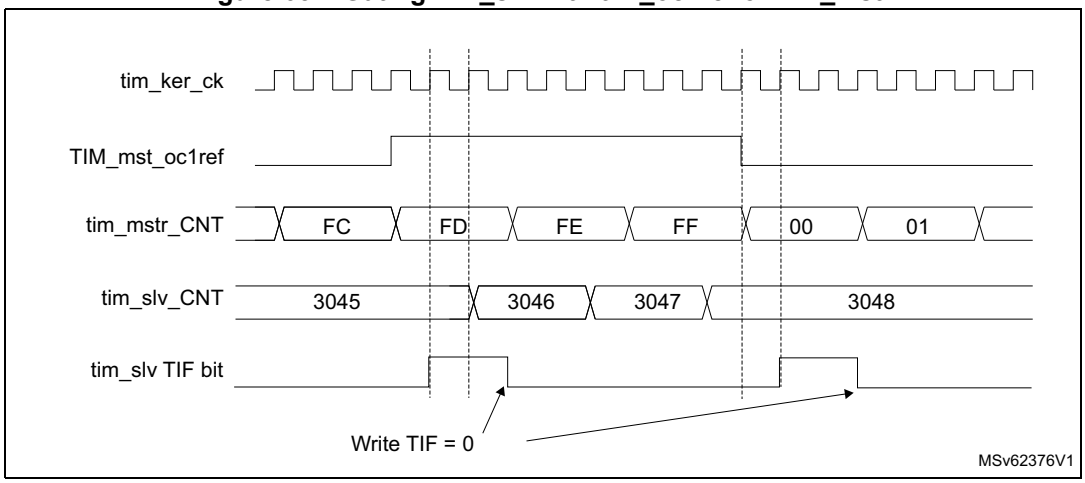
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 649* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 651. Gating TIM_slv with tim_oc1ref of TIM_mstr

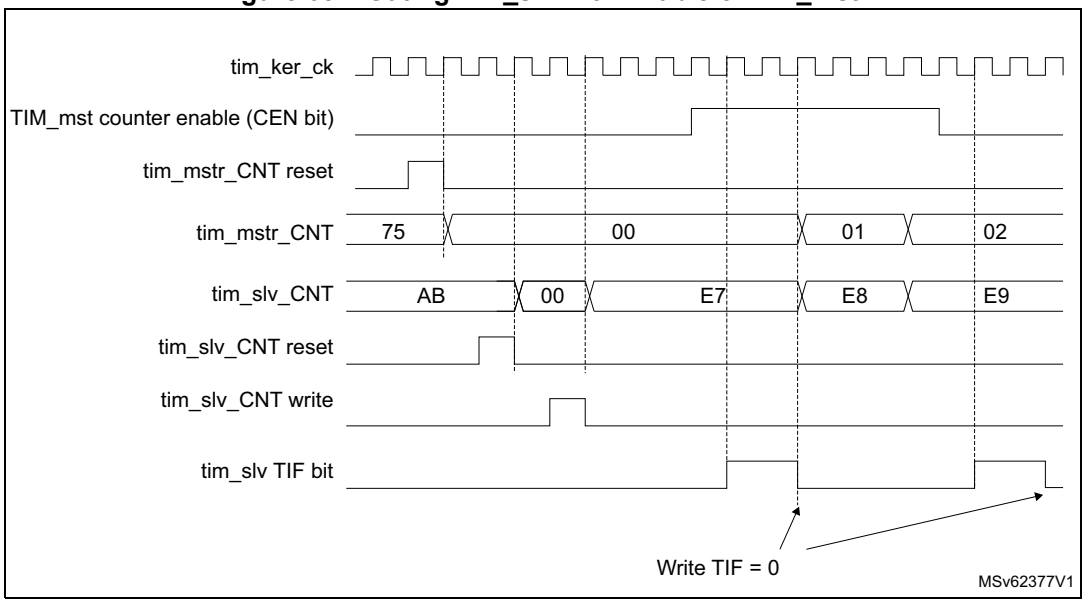


In the example in *Figure 651*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIM_EGR registers.

In the next example (refer to *Figure 652*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing 0xE7 in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 652. Gating TIM_slv with Enable of TIM_mstr

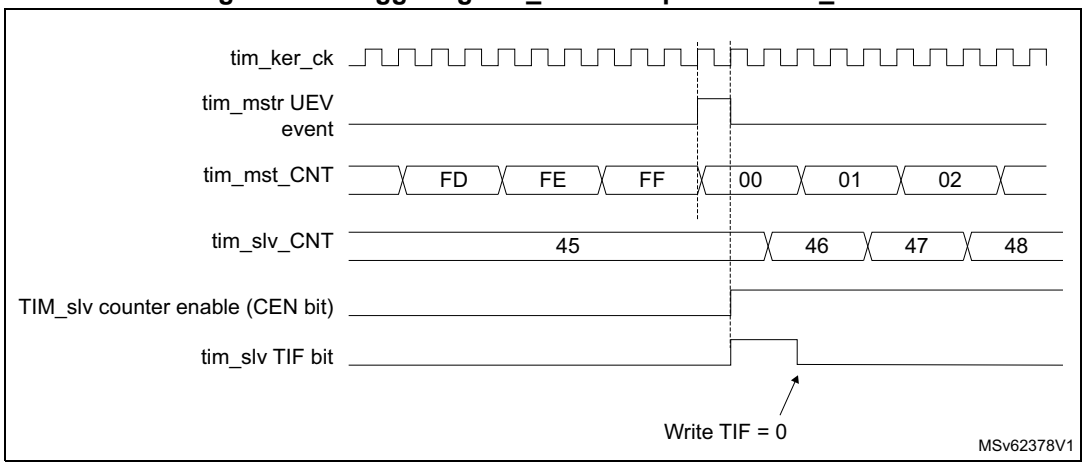


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 649* for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

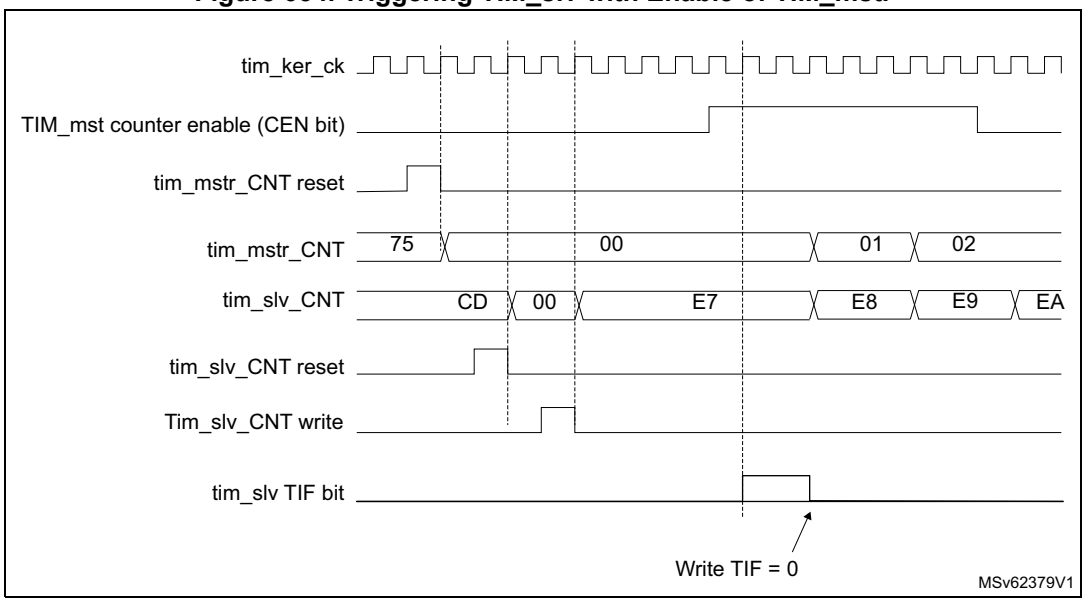
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Figure 653. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 654* shows the behavior with the same configuration as in *Figure 653* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 654. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

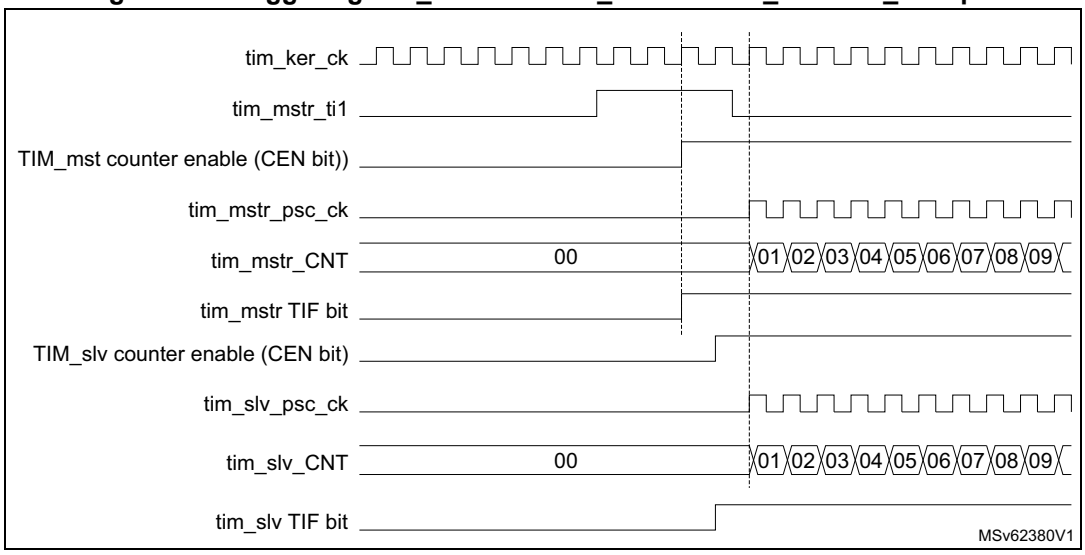
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 649* for connections. To ensure the counters are aligned, TIM_mstr must be configured in master/slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in master/slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIM_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 655. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.4.28 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.4.29 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period.

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period.

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

42.4.30 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIM_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register:

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 42.5.23: TIM DMA control register (TIM_DCR)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

42.4.31 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 491*.

Table 491. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trgi_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

42.4.32 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

42.4.33 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 492. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

42.4.34 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 493*.

Table 493. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

42.5 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

42.5.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (tDTS) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: $t_{DTS} = 8 \times t_{tim_ker_ck}$

Bit 7 **ARPE**: Autoreload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in center-aligned mode or encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

42.5.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI3INV	TI2INV	TI1INV	AD SYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw		rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bit 31 **TI3INV**: tim_ti3 signal inversion on XOR gate input
0: tim_ti3 non-inverted
1: tim_ti3 inverted

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input
0: tim_ti2 non-inverted
1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input
0: tim_ti1 non-inverted
1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization
0: The timer operates independently from the ADC
1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 42.4.29* for requirements.
The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection
This bit defines the position of the XOR gate with respect to input filters.
0: XOR gate placed before TI1 filter
1: XOR gate placed after TI1, TI2 and TI3 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection
This bit enables the insertion of a XOR gate to combine tim_ti1_in[15:0], tim_ti2_in[15:0], and tim_ti3_in[15:0] inputs into tim_ti1.
The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.
0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 41.3.34: Interfacing with Hall sensors*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

42.5.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	SMS[4:3]	
		rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:26 **SETPS[3:0]**: Synchronous external trigger prescaler

The linear synchronous prescaler divides the frequency of the tim_etrf signal, by a factor (SETPS[3:0] + 1).

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[4:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[4:0] bitfield is preloaded

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables external clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etrp frequency divided by 2
10: tim_etrp frequency divided by 4
11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM**: Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source

0: tim_ocref_clr_int is connected to the tim_ocref_clr input

1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 42.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 17:16, 2:0 **SMS[4:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010: Encoder mode: Clock plus direction, x2 mode.

01011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

01100: Encoder mode: Directional clock, x2 mode.

01101: Encoder mode: Directional clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000: Quadrature encoder with built-in debouncer: x2 mode.

10001: Quadrature encoder with built-in debouncer: x4 mode.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.5.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

42.5.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
r	r	r	r				rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 31 **TI4FS**: tim_ti4f status
This flag indicates the value of the tim_ti4f signal (after the digital filtering stage), for polling purpose.
0: tim_ti4f signal is low
1: tim_ti4f signal is high

Bit 30 **TI3FS**: tim_ti3f status
This flag indicates the value of the tim_ti3f signal (after the digital filtering stage), for polling purpose.
0: tim_ti3f signal is low
1: tim_ti3f signal is high

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are three possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS = 0 in the TIM_CR1 register.
When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

42.5.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (up-counting), else it takes the autoreload value (TIM_ARR) if DIR = 1 (down-counting).

42.5.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000:No filter, sampling is done at f_{DTS}

0001:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100:$f_{SAMPLING} = f_{DTS}/2$, N = 6

0101:$f_{SAMPLING} = f_{DTS}/2$, N = 8

0110:$f_{SAMPLING} = f_{DTS}/4$, N = 6

0111:$f_{SAMPLING} = f_{DTS}/4$, N = 8

1000:$f_{SAMPLING} = f_{DTS}/8$, N = 6

1001:$f_{SAMPLING} = f_{DTS}/8$, N = 8

1010:$f_{SAMPLING} = f_{DTS}/16$, N = 5

1011:$f_{SAMPLING} = f_{DTS}/16$, N = 6

1100:$f_{SAMPLING} = f_{DTS}/16$, N = 8

1101:$f_{SAMPLING} = f_{DTS}/32$, N = 5

1110:$f_{SAMPLING} = f_{DTS}/32$, N = 6

1111:$f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

42.5.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M [4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M [4:3]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

0: tim_oc2ref is not affected by the tim_ocref_clr_int signal

1: tim_oc2ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 42.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode

Refer to OC1M description on bits 17:16, 6:4.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Note: This mode is not available for all PWM modes. Refer to Section 42.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - In up-counting, channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIM_CNT>TIM_CCR1 else active (tim_oc1ref = 1).

00111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active. In down-counting, channel 1 is active as long as TIM_CNT>TIM_CCR1 else inactive.

01000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Reserved.

01011: Reserved.

01100-01101: Combined PWM mode 1 and 2 - Refer to *Table 488: Combined PWM mode overview* for details.

01110-01111: Asymmetric PWM mode 1 and 2 - Refer to *Table 487: Asymmetric PWM modes* for details.

Others: Reserved.

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to three clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

42.5.9 TIM capture/compare mode register 2 (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

42.5.10 TIM capture/compare mode register 2 [alternate] (TIM_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable
0: tim_oc4ref is not affected by the tim_ocref_clr_int signal
1: tim_oc4ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)
Note: This mode is not available for all PWM modes. Refer to Section 42.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 25:24, 14:12 **OC4M[4:0]**: Output compare 4 mode
Refer to OC3M[4:0].

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIM_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

0: tim_oc3ref is not affected by the tim_ocref_clr_int signal
1: tim_oc3ref is cleared as soon as a high level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Note: This mode is not available for all PWM modes. Refer to Section 42.4.16: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC3M[4:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR3 and the counter TIM_CNT has no effect on the outputs.(this mode is used to generate a timing base).

00001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIM_CNT matches the capture/compare register 3 (TIM_CCR3).

00011: Toggle - tim_oc3ref toggles when TIM_CNT = TIM_CCR3.

00100: Force inactive level - tim_oc3ref is forced low.

00101: Force active level - tim_oc3ref is forced high.

00110: PWM mode 1 - In up-counting, channel 3 is active as long as TIM_CNT<TIM_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIM_CNT>TIM_CCR3 else active (tim_oc3ref = 1).

00111: PWM mode 2 - In up-counting, channel 3 is inactive as long as TIM_CNT<TIM_CCR3 else active. In down-counting, channel 3 is active as long as TIM_CNT>TIM_CCR3 else inactive.

01000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Pulse on compare - A pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIM_ECR.

01011: Direction output - The tim_oc3ref signal is overridden by a copy of the DIR bit.

01100-01101: Combined PWM mode 1 and 2 - Refer to *Table 488: Combined PWM mode overview* for details.

01110-01111: Asymmetric PWM mode 1 and 2 - Refer to *Table 487: Asymmetric PWM modes* for details.

10110-11001: Asymmetric PWM mode 7 to 10 - Available on OC3M only. Refer to *Table 487: Asymmetric PWM modes* for details.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIM_CCER).

42.5.11 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: Capture/compare 4 output Polarity.
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/compare 4 output Polarity.
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable.
refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 output Polarity.
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/compare 3 output Polarity.
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable.
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 output Polarity.
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/compare 2 output Polarity.
refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable.
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/compare 1 output Polarity.

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1:non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:this configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/compare 1 output enable.

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 494. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.

42.5.12 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY_CNT[31]	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY_CNT[31]**: Value depends on IUFREMAP in TIM_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIM_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

42.5.13 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

42.5.14 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARR[31:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 42.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

42.5.15 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

42.5.16 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

42.5.17 TIM capture/compare register 3 (TIM_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIM_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

42.5.18 TIM capture/compare register 4 (TIM_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIM_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

42.5.19 TIM timer encoder control register (TIM_ECR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[4:0] = 00001, 00010, 00011, 01110, 01111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[4:0] = 01010, 01011, 01100, 01101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

42.5.20 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

42.5.21 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[4:2]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:14 **ETRSEL[4:0]**: etr_in source selection
These bits select the etr_in input source.
00000: tim_etr0: TIM_ETR input
00001: tim_etr1
...
11111: tim_etr31
Refer to *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

42.5.22 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
0000: tim_ocref_clr0
0001: tim_ocref_clr1
...
0111: tim_ocref_clr15
Refer to *Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

42.5.23 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11111: 32 transfers

Example: Let us consider the following transfer: DBL = 6 bytes & DBA = TIM2_CR1.

–If DBL = 6 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation: (TIM_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIM_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIM_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:
00000: TIM_CR1,
00001: TIM_CR2,
00010: TIM_SMCR,
...

42.5.24 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

42.5.25 TIM2/TIM3/TIM4/TIM5 register map

TIM2/TIM3/TIM4/TIM5 registers are mapped as described in the table below.

Table 495. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0	0	0	0	0	0	0	0
0x004	**TIM_CR2**	TI3INV	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value	0	0	0	0	0		0																		0	0	0	0	0			
0x008	**TIM_SMCR**	Res.	Res.	SETPS[3:0]				SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS [4:3]		ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value			0	0	0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0		0	0	0	0	0	0	0		0	0	0	0	0
0x010	**TIM_SR**	TI4FS	TI3FS	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value	0	0	0	0				0	0	0	0	0								0	0	0	0			0		0	0	0	0	0
0x014	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	**TIM_CCMR1** Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 495. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	**TIM_CCMR2** Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR2 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC3M[4:3]		O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x024	**TIM_CNT**	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**TIM_ARR**	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved																																
0x034	**TIM_CCR1**	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIM_CCR2**	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**TIM_CCR3**	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	**TIM_CCR4**	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	**TIM_ECR**	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	**TIM_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	**TIM_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [4:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0	0	0														

Table 495. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x064	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

43 Basic timers (TIM6/TIM7/TIM26/TIM27)

43.1 TIM6/TIM7/TIM26/TIM27 introduction

The basic timers TIM6/TIM7/TIM26/TIM27 consist in a 16-bit autoreload counter driven by a programmable prescaler.

They can be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

43.2 TIM6/TIM7/TIM26/TIM27 main features

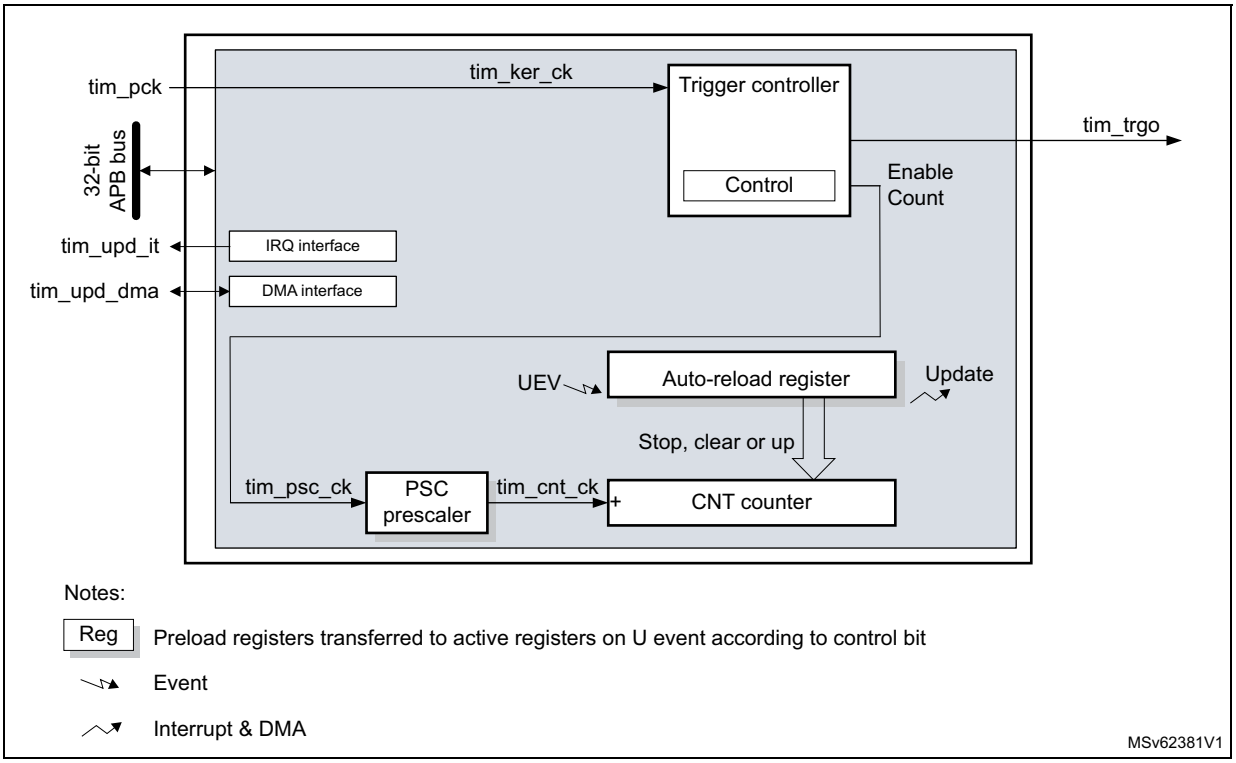
Basic timer (TIM6/TIM7/TIM26/TIM27) features include:

- 16-bit autoreload upcounter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Synchronization circuit to trigger the DAC.
- Interrupt/DMA generation on the update event: counter overflow.
- ADC synchronization for jitter-free sampling points.

43.3 TIM6/TIM7/TIM26/TIM27 functional description

43.3.1 TIM6/TIM7/TIM26/TIM27 block diagram

Figure 656. Basic timer block diagram



43.3.2 TIM6/TIM7/TIM26/TIM27 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 496. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

43.3.3 TIM6/TIM7/TIM26/TIM27 clocks

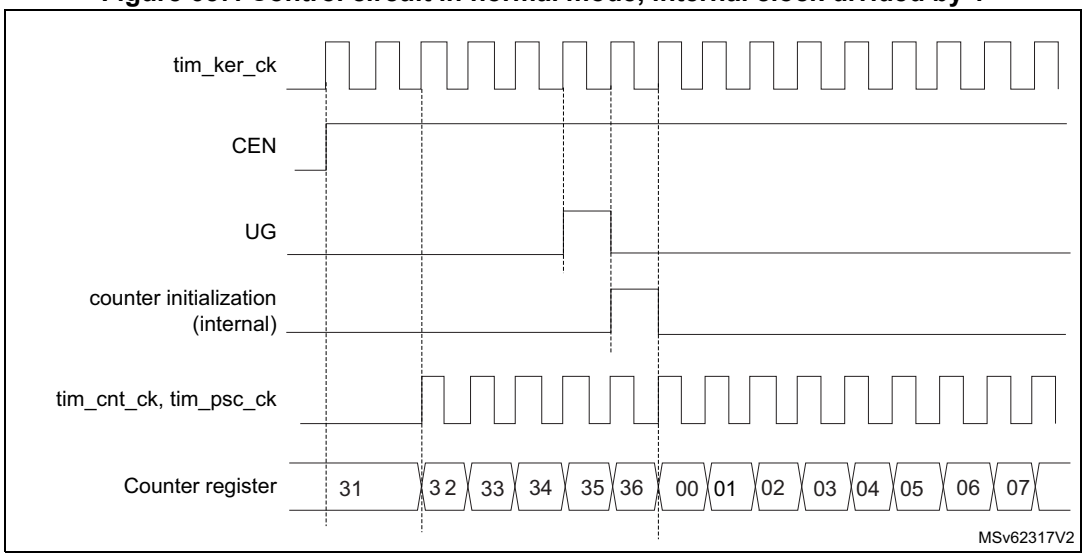
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 657 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 657. Control circuit in normal mode, internal clock divided by 1



43.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Auto-Reload register (TIM_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIM_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIM_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as the TIM_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 658 and *Figure 659* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 658. Counter timing diagram with prescaler division change from 1 to 2

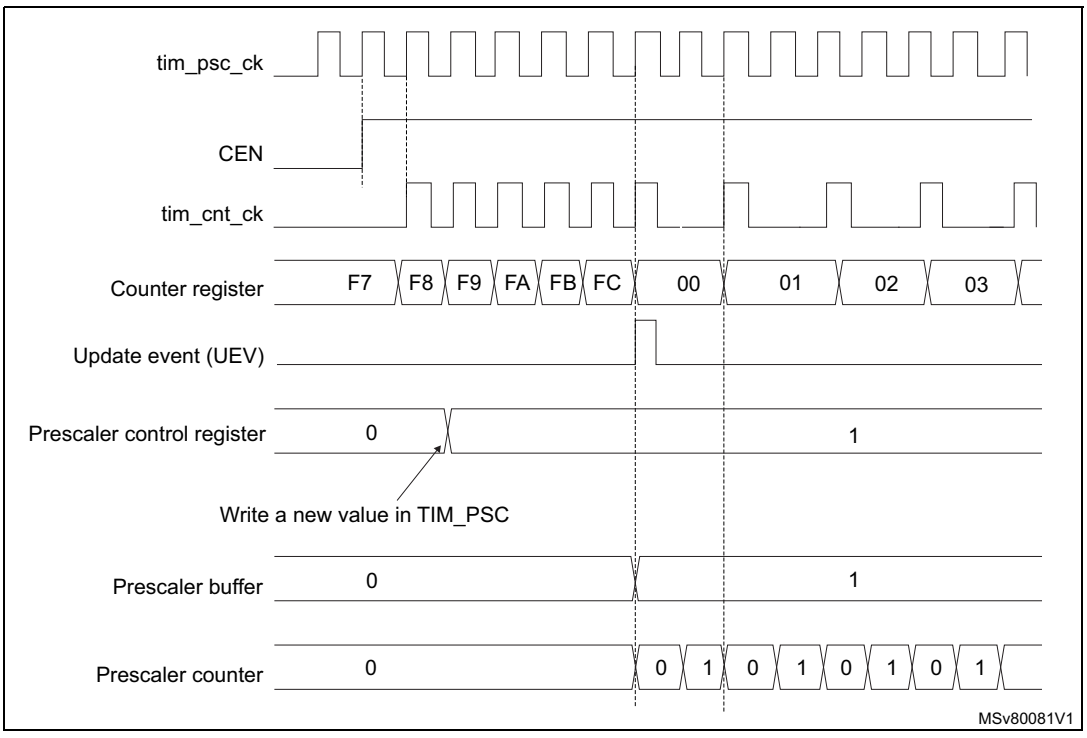
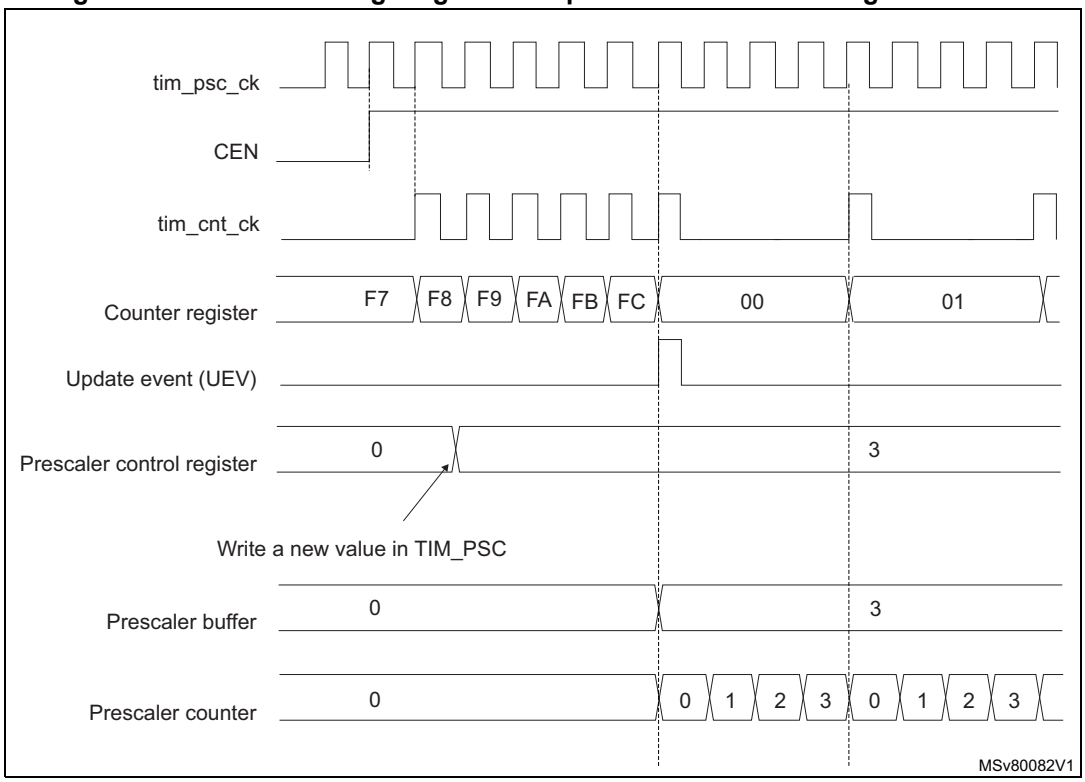


Figure 659. Counter timing diagram with prescaler division change from 1 to 4



43.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIM_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIM_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIM_PSC register).
- The autoreload shadow register is updated with the preload value (TIM_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 660. Counter timing diagram, internal clock divided by 1

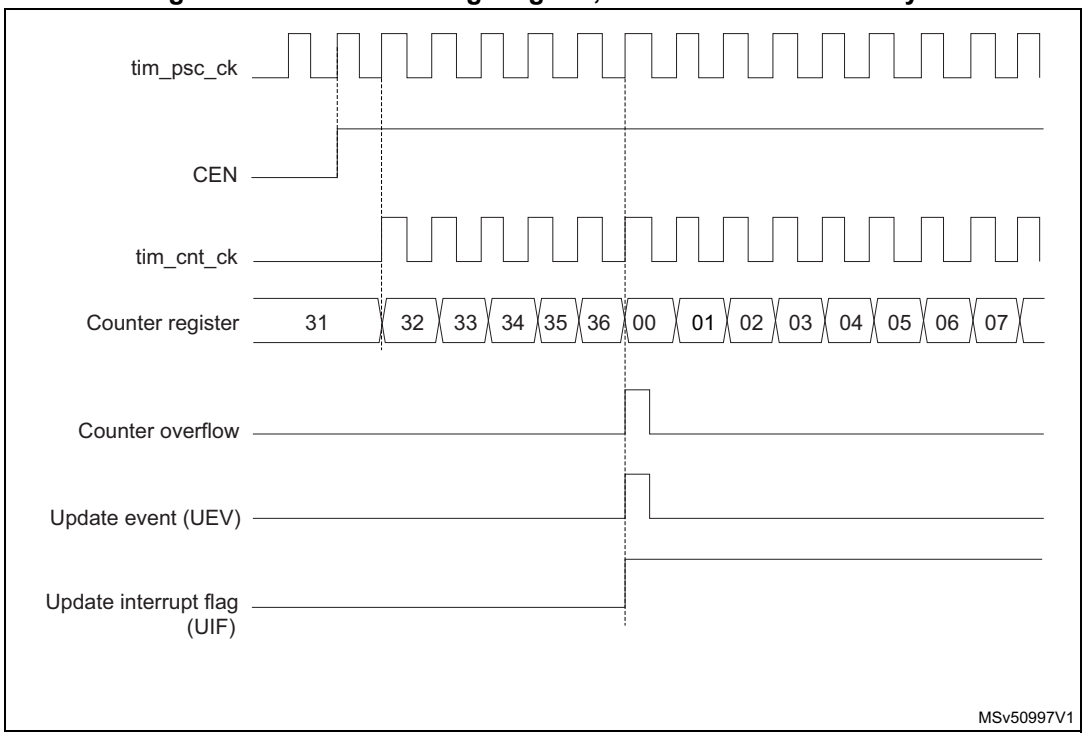


Figure 661. Counter timing diagram, internal clock divided by 2

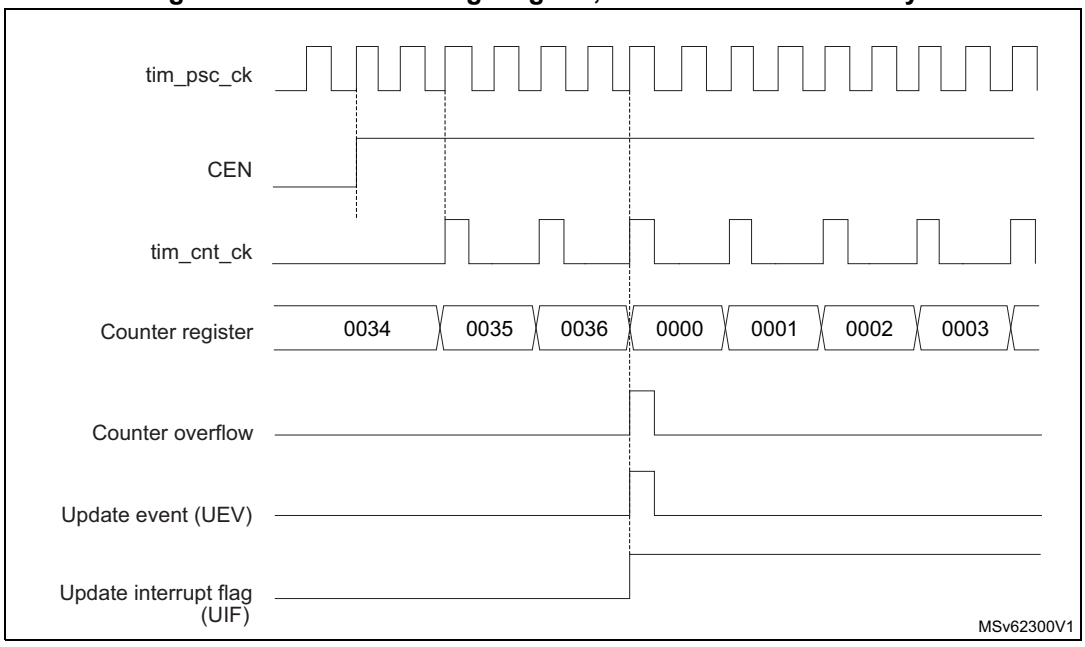


Figure 662. Counter timing diagram, internal clock divided by 4

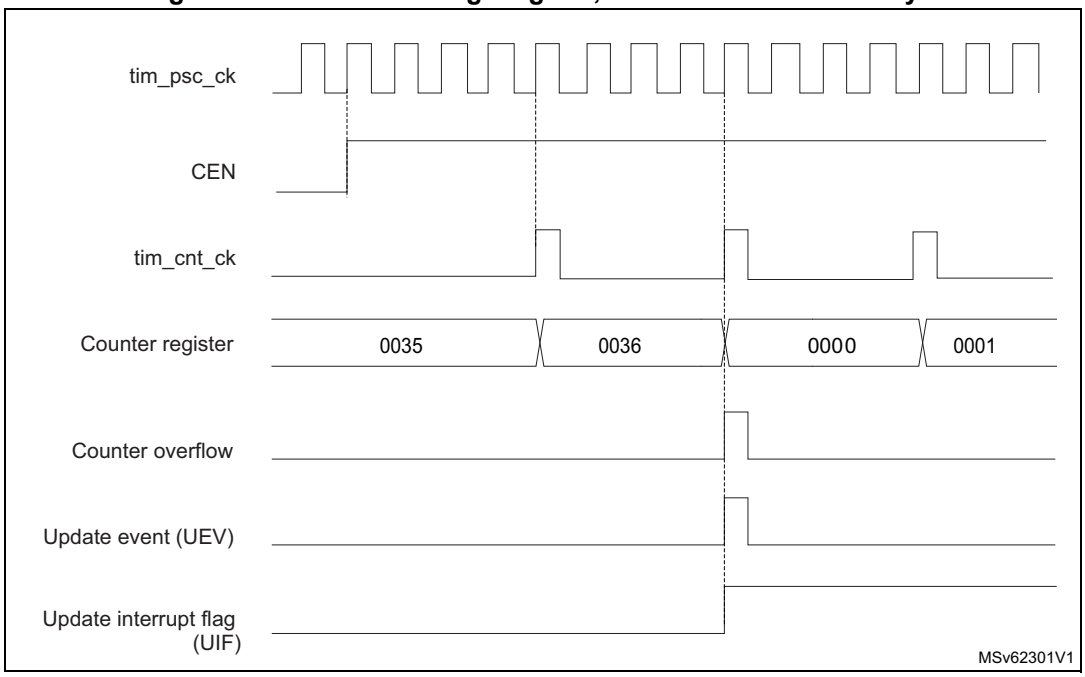


Figure 663. Counter timing diagram, internal clock divided by N

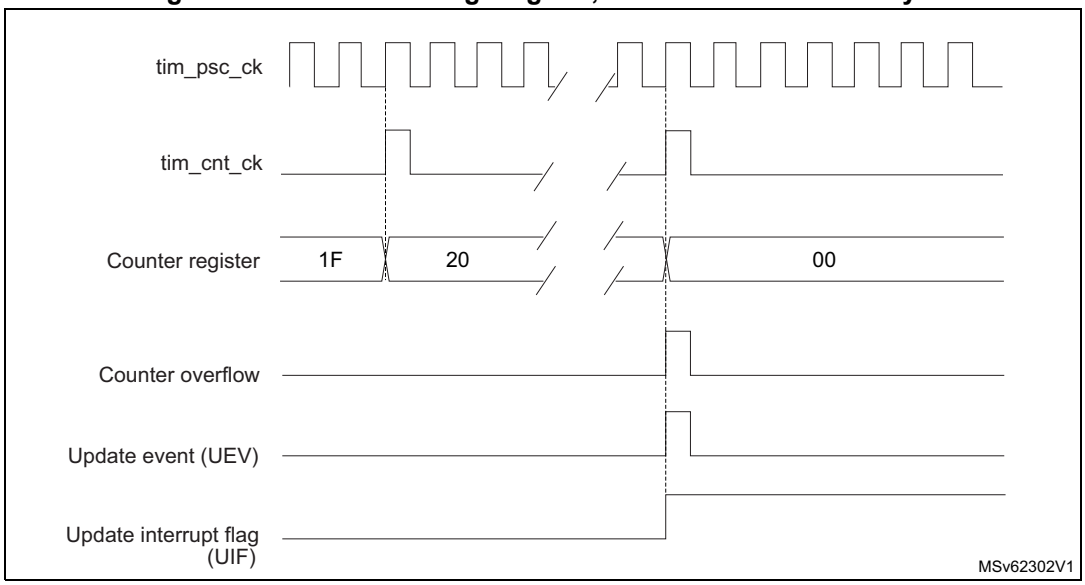


Figure 664. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

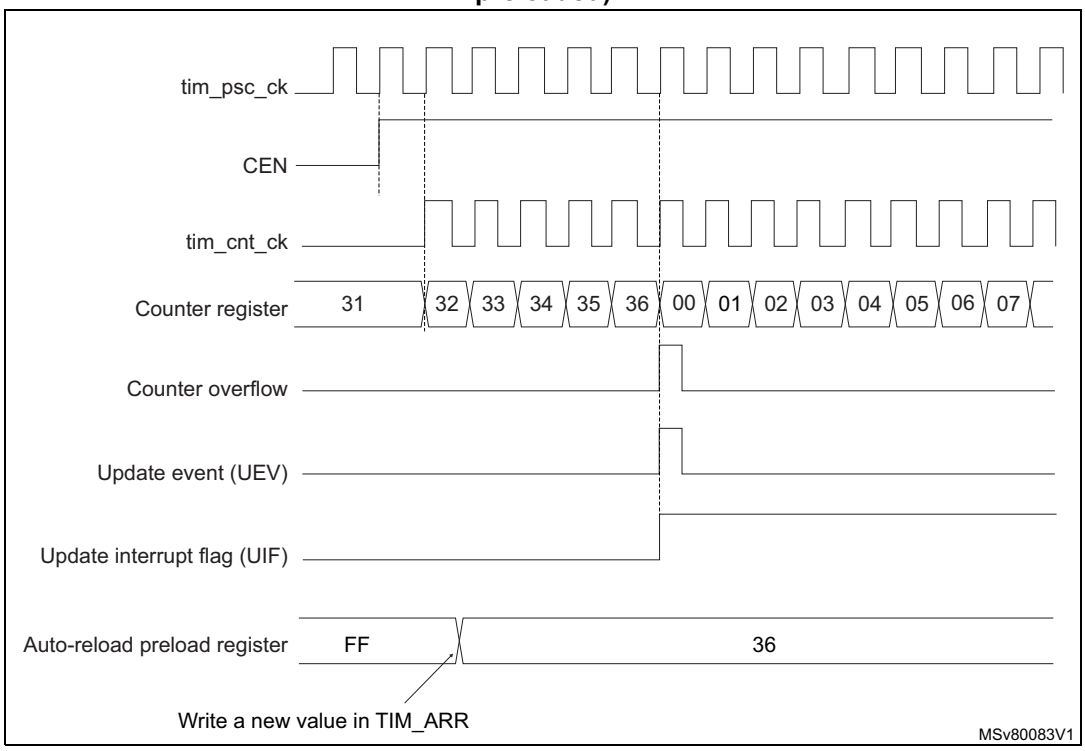
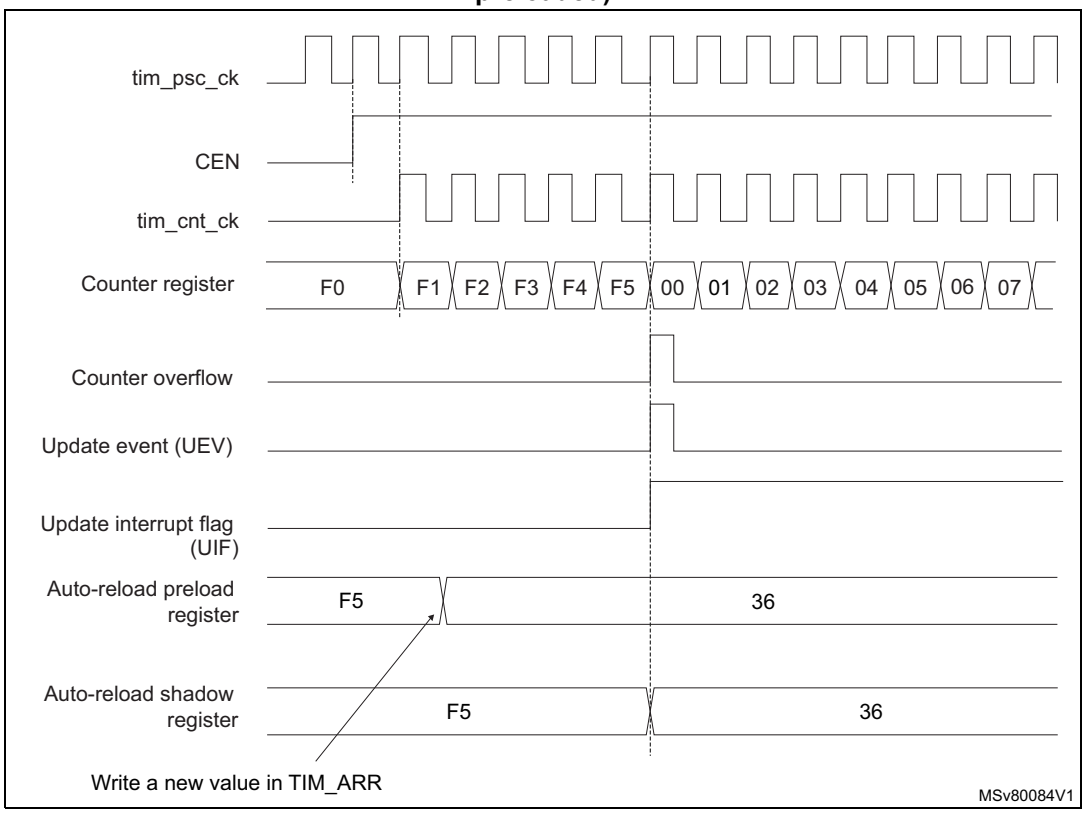


Figure 665. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



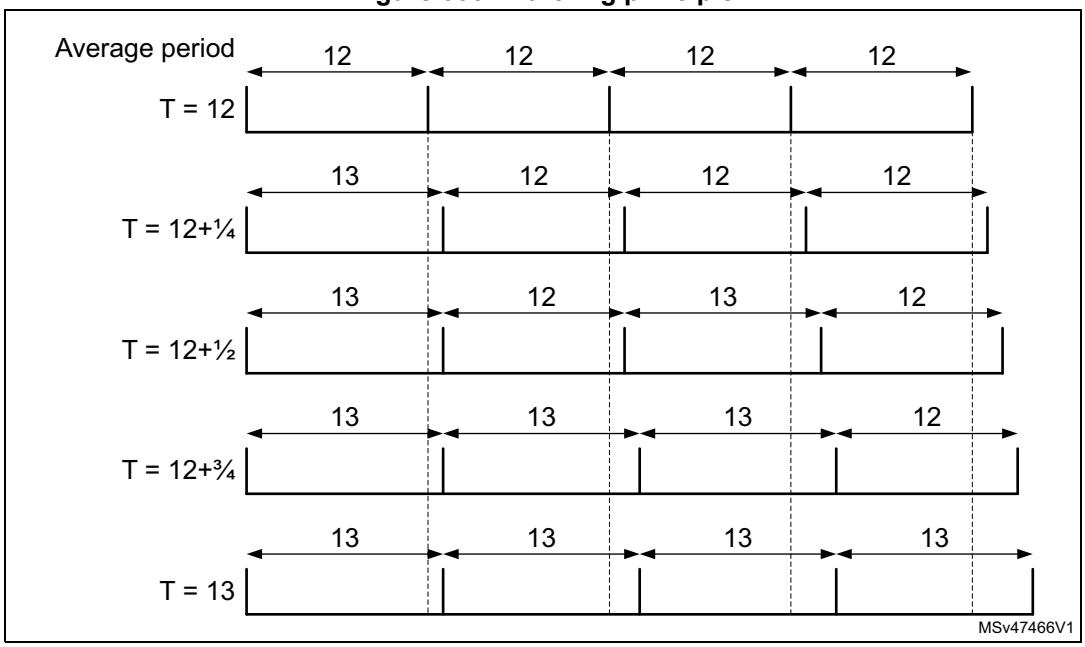
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This affects the way the TIM_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 666 presents the dithering principle applied to four consecutive counting periods.

Figure 666. Dithering principle

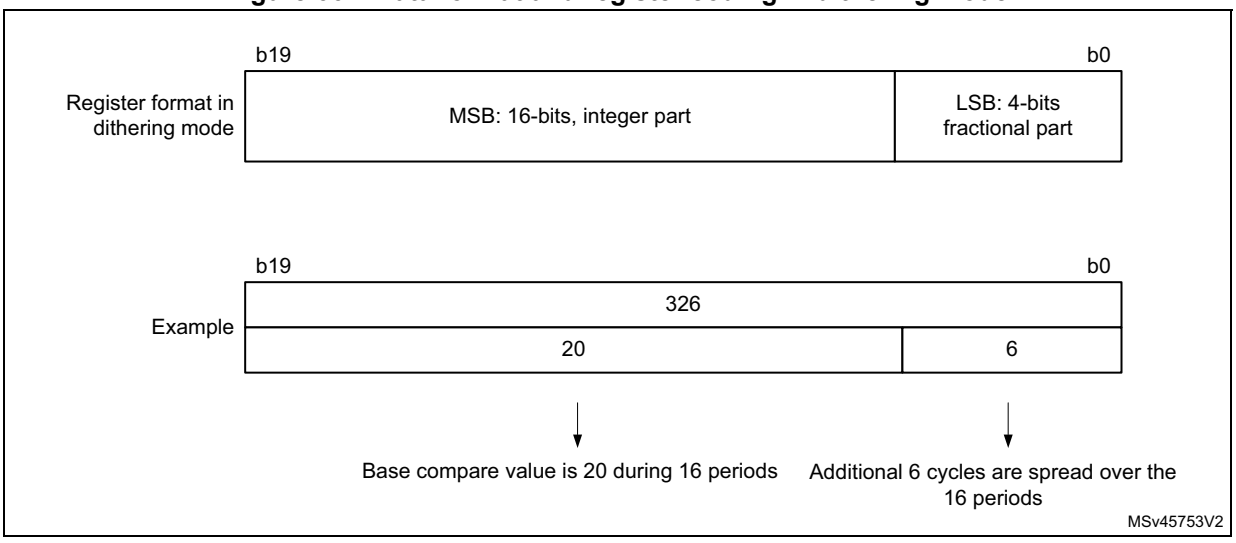


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 667* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

Figure 667. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

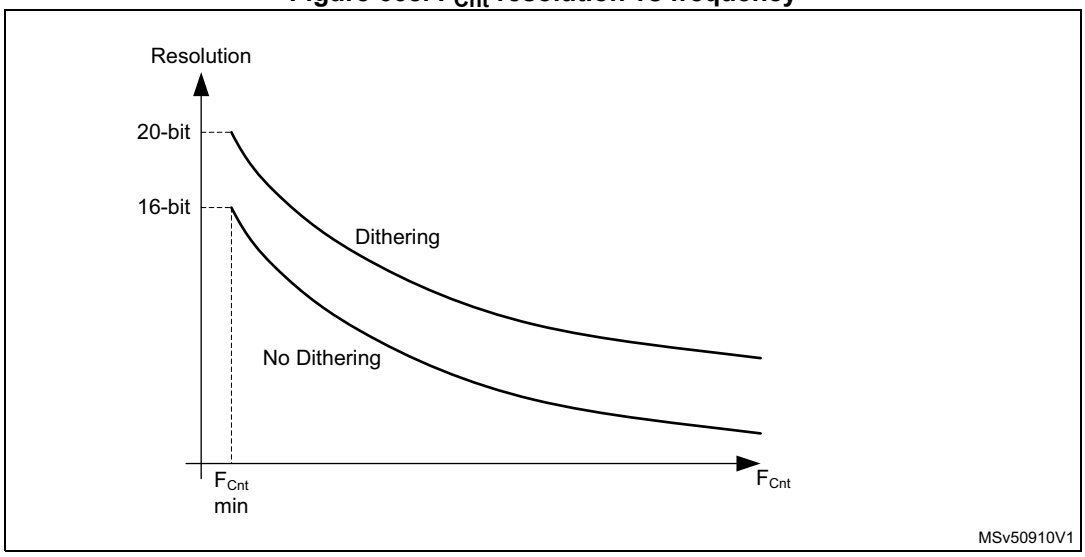
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

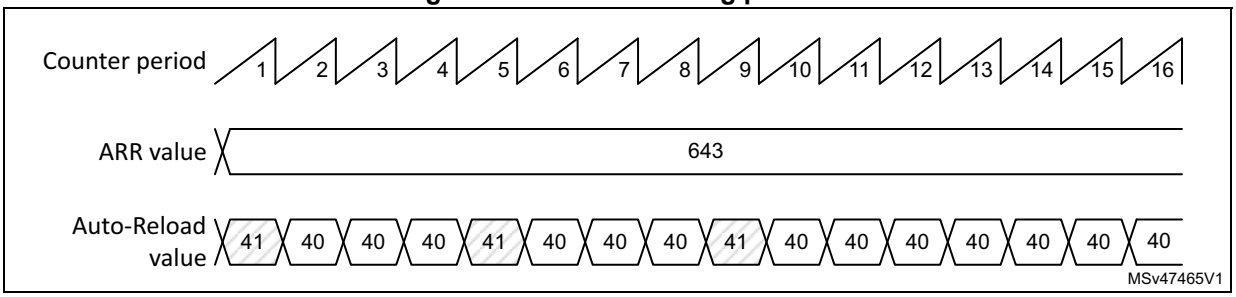
As shown on *Figure 668*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 668. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 669*.

Figure 669. PWM dithering pattern



The autoreload and compare values increments are spread following the specific patterns described in *Table 497*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 497. TIM_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

43.3.6 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

43.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

43.3.8 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

43.3.9 TIM6/TIM7/TIM26/TIM27 DMA requests

The TIM6/TIM7/TIM26/TIM27 can generate a single DMA request, as shown in *Table 498*.

Table 498. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE

43.3.10 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

43.3.11 TIM6/TIM7/TIM26/TIM27 low-power modes

Table 499. Effect of low-power modes on TIM6/TIM7/TIM26/TIM27

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

43.3.12 TIM6/TIM7/TIM26/TIM27 interrupts

The TIM6/TIM7/TIM26/TIM27 can generate a single interrupt, as shown in *Table 500*.

Table 500. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7 TIM26 TIM27	Update	UIF	UIE	write 0 in UIF	Yes	No

43.4 TIM6/TIM7/TIM26/TIM27 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

43.4.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM_ARR register is not buffered.
1: TIM_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

43.4.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 43.3.8* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000:**Reset** - the UG bit from the TIM_EGR register is used as a trigger output (tim_trgo).

001:**Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.

010:**Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

43.4.3 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

43.4.4 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.

0: No Update Interrupt Overrun

1: Update Interrupt Overrun

Bits 23:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

–On counter overflow if UDIS = 0 in the TIM_CR1 register.

–When CNT is reinitialized by software using the UG bit in the TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

43.4.5 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

43.4.6 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in TIM_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

43.4.7 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIM_EGR register.

43.4.8 TIM autoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 43.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

43.4.9 TIM6/TIM7/TIM26/TIM27 register map

TIM6/TIM7/TIM26/TIM27 registers are mapped as 16-bit addressable registers as described in the table below:

Table 501. TIM6/TIM7/TIM26/TIM27 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0				0				0	0	0	0
0x04	**TIM_CR2**	Res.	Res.	Res.	ADSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value				0																						0	0	0				
0x08	Reserved	Res.																															
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	**TIM_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value								0																								0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved	Res.																															
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

44 General-purpose timers (TIM9/TIM10/TIM11/TIM12/TIM13/TIM14)

44.1 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 introduction

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 44.4.21: Timer synchronization (TIM9/TIM12 only)*.

44.2 TIM9/TIM12 main features

The features of the TIM9/TIM12 general-purpose timers include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization, or count by internal trigger)
  - Input capture
  - Output compare
- ADC synchronization for jitter-free sampling points

44.3 TIM10/TIM11/TIM13/TIM14 main features

The features of general-purpose timers TIM10/TIM11/TIM13/TIM14 include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

44.4 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 functional description

44.4.1 Block diagram

Figure 670. General-purpose timer block diagram (TIM9/TIM12)

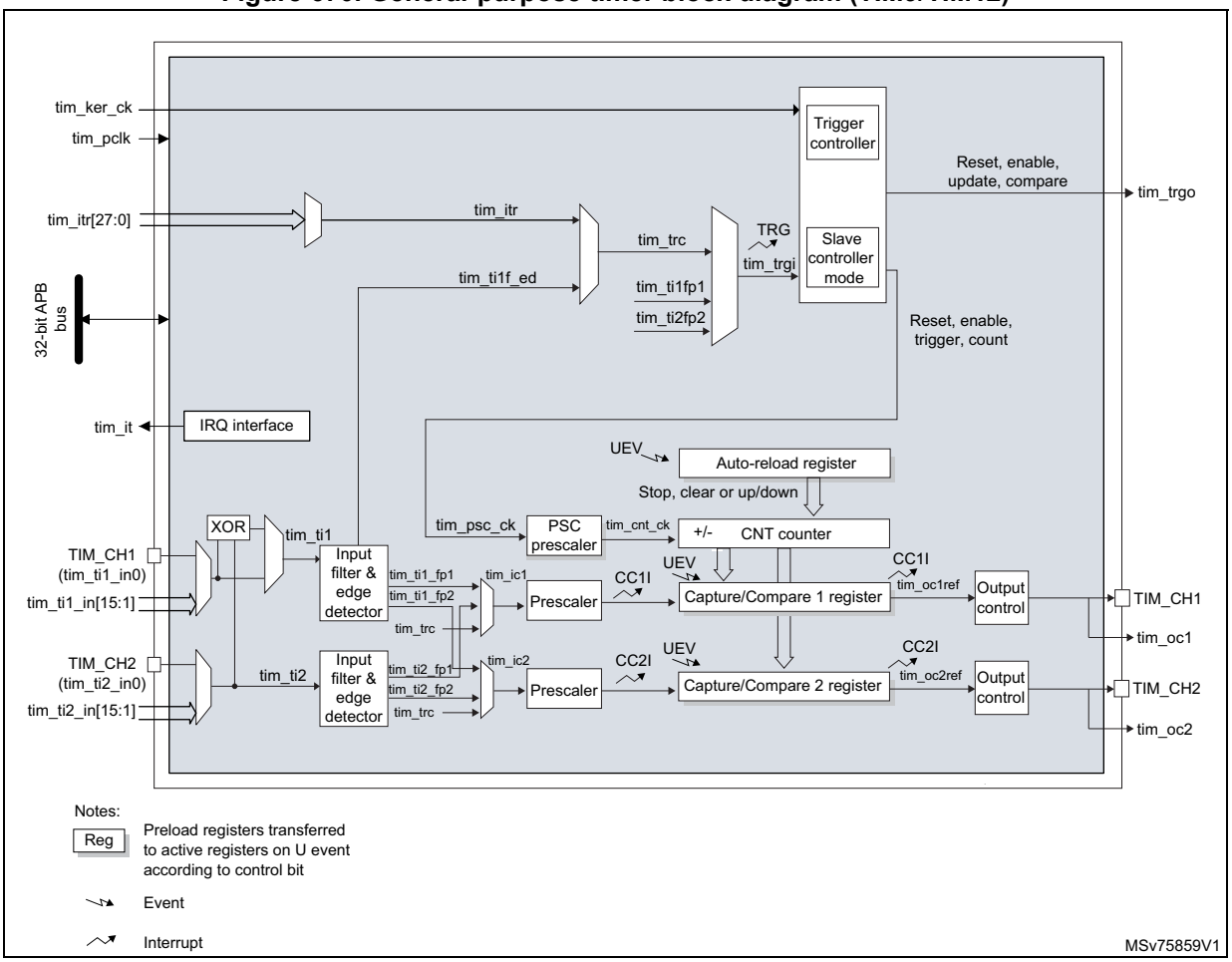
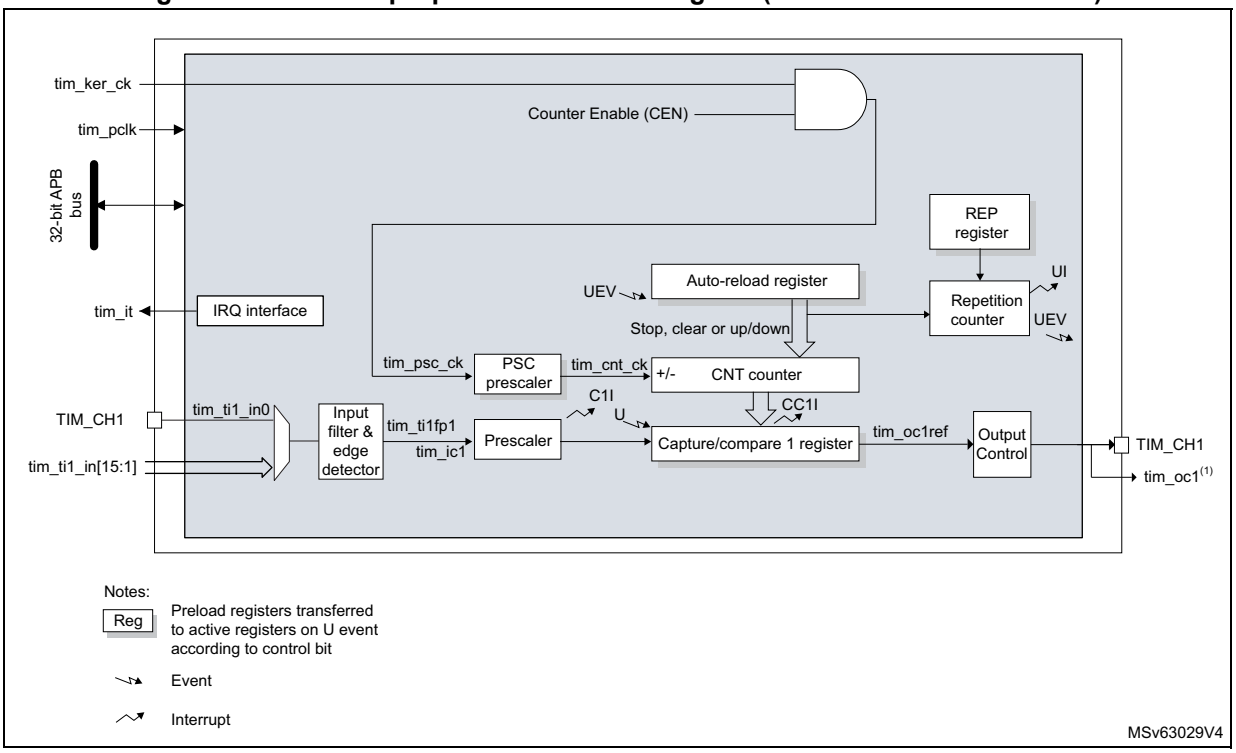


Figure 671. General-purpose timer block diagram (TIM10/TIM11/TIM13/TIM14)



1. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 44.4.22: Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only)* for details.

44.4.2 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals

Table 502 and *Table 503* in this section summarize the TIM inputs and outputs.

Table 502. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.

1. Available for TIM9/TIM12 only.

Table 503. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[27:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_oc1 tim_oc2(1)	Output	Internal timer output. Can be used for triggering other timers or the ADC(s).

Table 503. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, and trigger requests

1. Available for TIM9/TIM12 only.

Table 504 and *Table 505* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 504. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources					
	TIM9	TIM10	TIM11	TIM12	TIM13	TIM14
tim_ti1_in0	TIM9_CH1	TIM10_CH1	TIM11_CH1	TIM12_CH1	TIM13_CH1	TIM14_CH1
tim_ti1_in1	comp1_out	comp3_out	comp4_out	comp5_out	comp7_out	comp8_out
tim_ti1_in2	rcc_mco1	i3c1_ibi_ack	Reserved	rcc_mco1	i3c1_ibi_ack	Reserved
tim_ti1_in3	rcc_mco2	Reserved		rcc_mco2	Reserved	
tim_ti1_in4	hsi_cal_ck			hsi_cal_ck		
tim_ti1_in5	Reserved			Reserved		
tim_ti1_in[6]	usb_sof			usb_sof		
tim_ti1_in[7]	tim1_trgo			tim4_trgo		
tim_ti1_in[8]	tim2_trgo			tim20_trgo		
tim_ti1_in[9]	play1_out13(1)	play1_out10(1)	play1_out11(1)	play1_out7(1)	play1_out12(1)	play1_out1(1)
tim_ti1_in[15:10]	Reserved					

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability.

Table 505. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM9	TIM12
tim_ti2_in0	TIM9_CH2	TIM12_CH2
tim_ti2_in1	comp2_out	comp6_out

Table 505. Interconnect to the tim_ti2 input multiplexer (continued)

tim_ti2 inputs	Sources	
	TIM9	TIM12
tim_ti2_in2	Reserved	Reserved
tim_ti2_in3		
tim_ti2_in4		
tim_ti2_in5		
tim_ti2_in6	tim15_trgo	tim15_trgo
tim_ti2_in7	tim3_trgo	tim5_trgo
tim_ti2_in8	tim8_trgo	Reserved
tim_ti2_in9	play1_out1(1)	play1_out14(1)
tim_ti2_in[15:10]	Reserved	

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability.

Table 506 lists the internal sources connected to the tim_itr input multiplexer.

Table 506. TIMx internal trigger connection

tim_itrx inputs	TIM9	TIM12
tim_itr0	tim1_trgo	tim1_trgo
tim_itr1	tim1_trgo2	tim1_trgo2
tim_itr2	tim2_trgo	tim2_trgo
tim_itr3	tim3_trgo(1)	tim3_trgo(1)
tim_itr4	tim4_trgo(1)	tim4_trgo(1)
tim_itr5	tim5_trgo(1)	tim5_trgo(1)
tim_itr6	tim8_trgo	tim8_trgo
tim_itr7	tim8_trgo2	tim8_trgo2
tim_itr8	Reserved	tim9_trgo
tim_itr9	tim10_oc1	tim10_oc1
tim_itr10	tim11_oc1	tim11_oc1
tim_itr11	tim12_trgo	Reserved
tim_itr12	tim13_oc1	tim13_oc1
tim_itr13	tim14_oc1	tim14_oc1
tim_itr14	tim15_trgo	tim15_trgo
tim_itr15	tim16_oc1	tim16_oc1
tim_itr16	tim17_oc1	tim17_oc1
tim_itr17	tim20_trgo	tim20_trgo
tim_itr18	tim20_trgo2	tim20_trgo2

Table 506. TIMx internal trigger connection (continued)

tim_itrx inputs	TIM9	TIM12
tim_itr19	Reserved	Reserved
tim_itr20		
tim_itr21	hrtim1_out_sync2	
tim_itr[27:22]	Reserved	Reserved

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability.

44.4.3 Time-base unit

The main block of the timer is a 16-bit up-counter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR).

The autoreload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- *The update interrupt service routine is too long with respect to the interval between consecutive update events.*
- *The update interrupt service routine was served too late with respect to the subsequent update event.*

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 672 and *Figure 673* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 672. Counter timing diagram with prescaler division change from 1 to 2

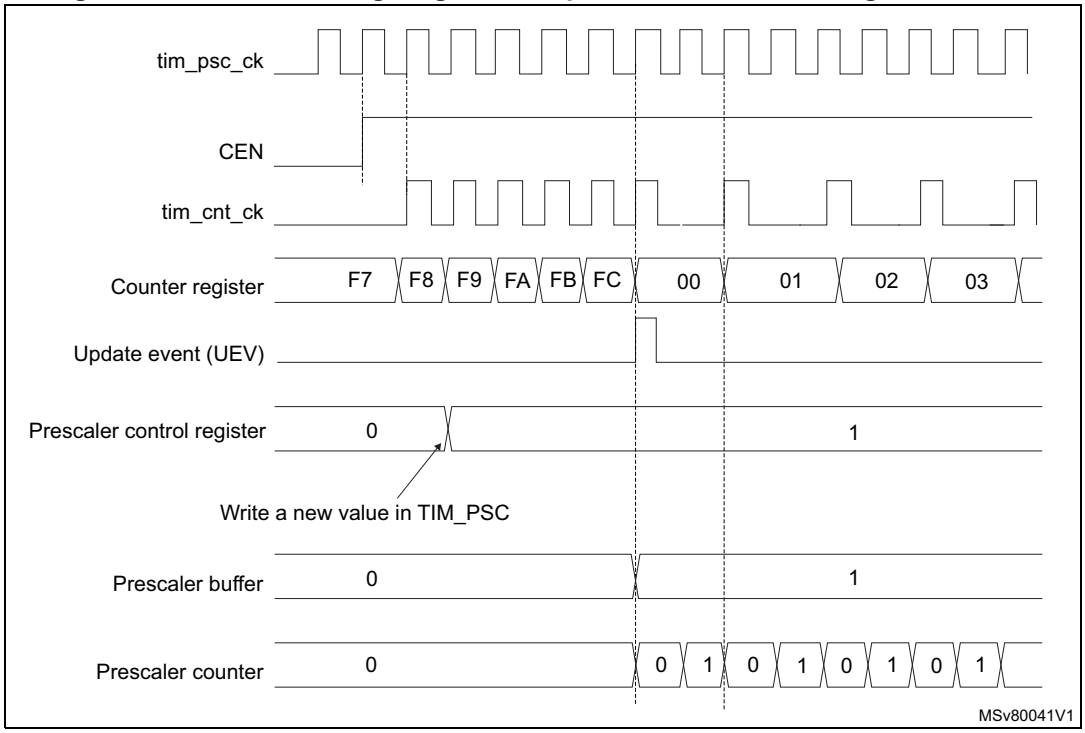
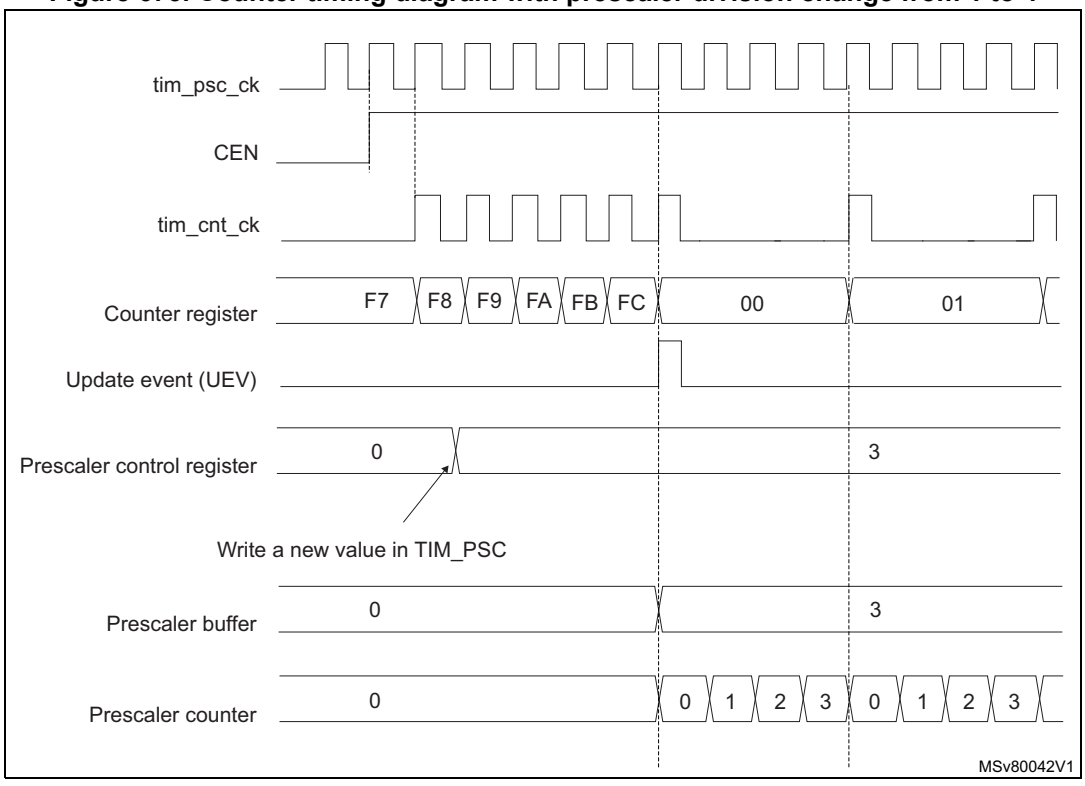


Figure 673. Counter timing diagram with prescaler division change from 1 to 4



44.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller on TIM9/TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The autoreload shadow register is updated with the preload value (TIM_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 674. Counter timing diagram, internal clock divided by 1

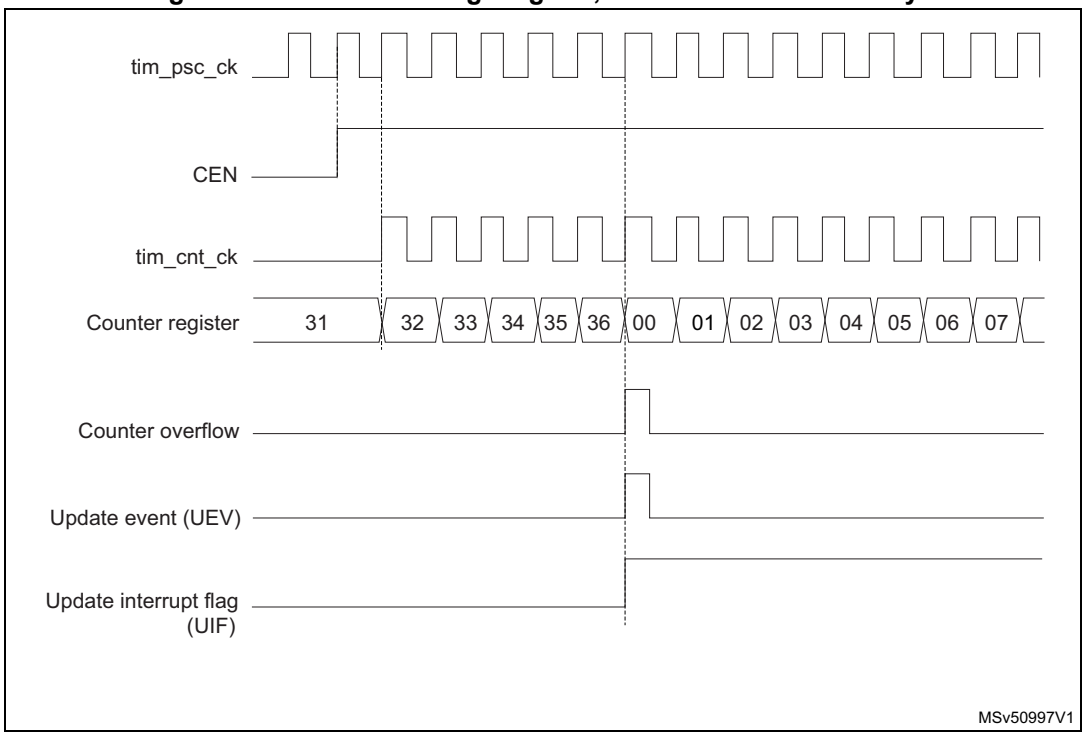


Figure 675. Counter timing diagram, internal clock divided by 2

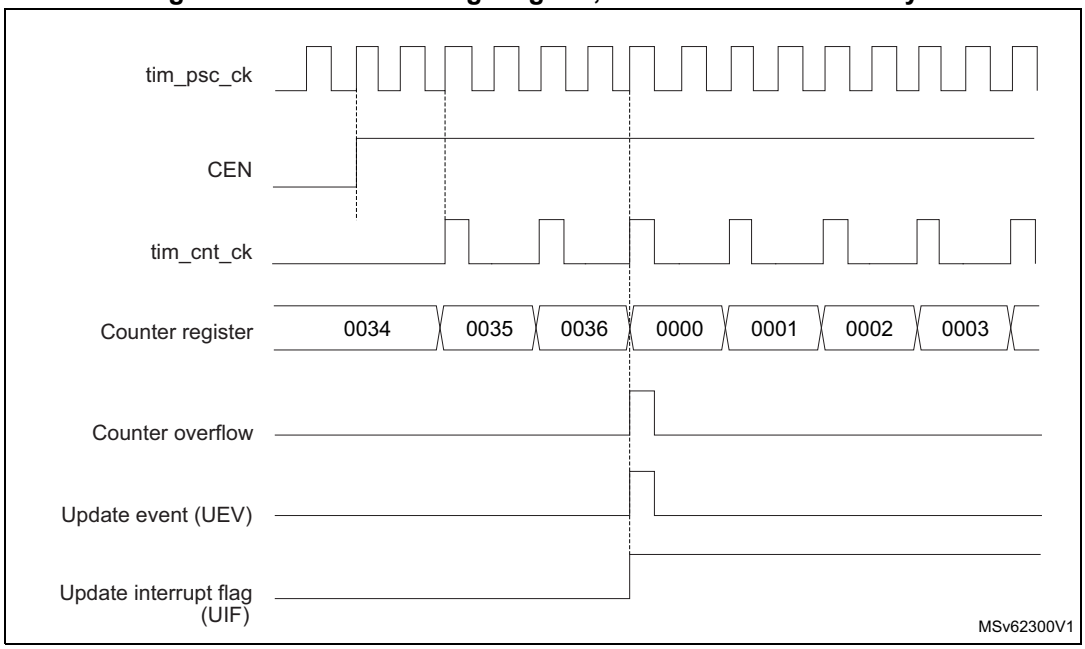


Figure 676. Counter timing diagram, internal clock divided by 4

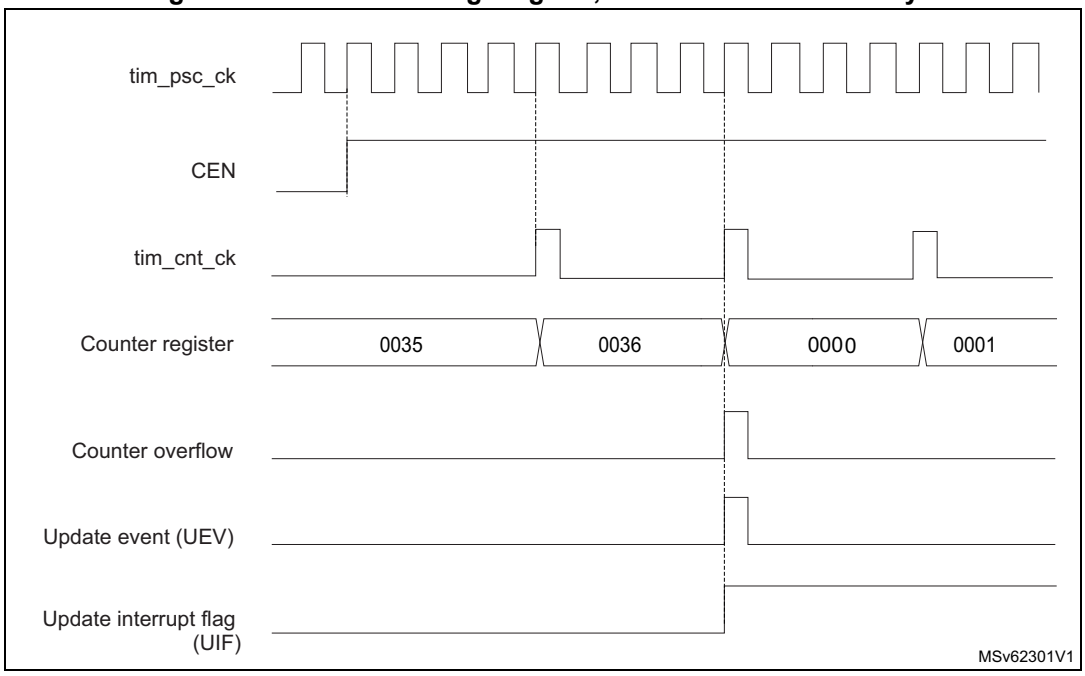


Figure 677. Counter timing diagram, internal clock divided by N

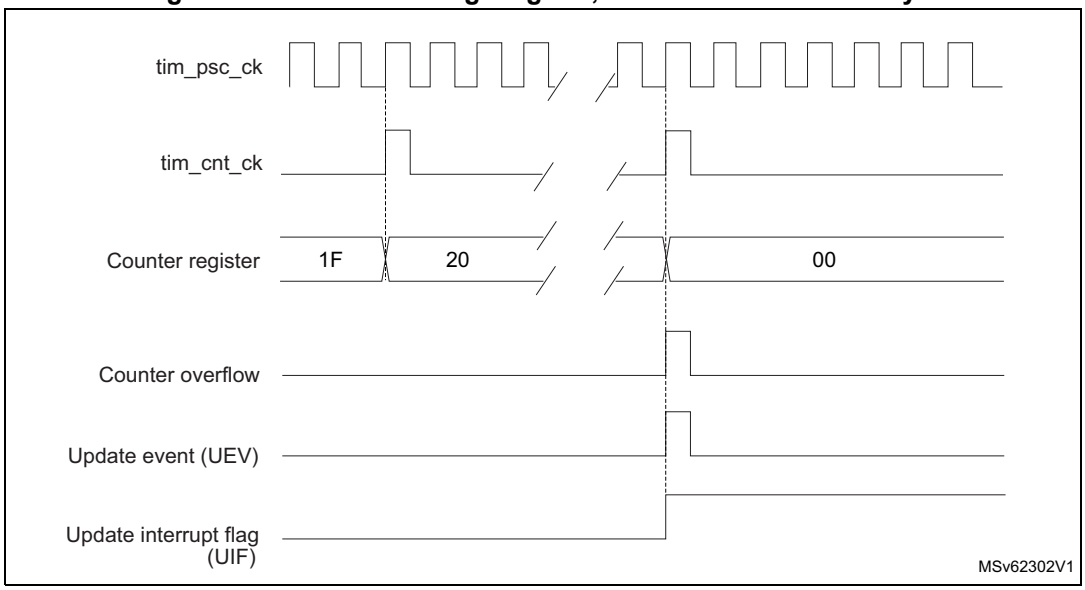


Figure 678. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

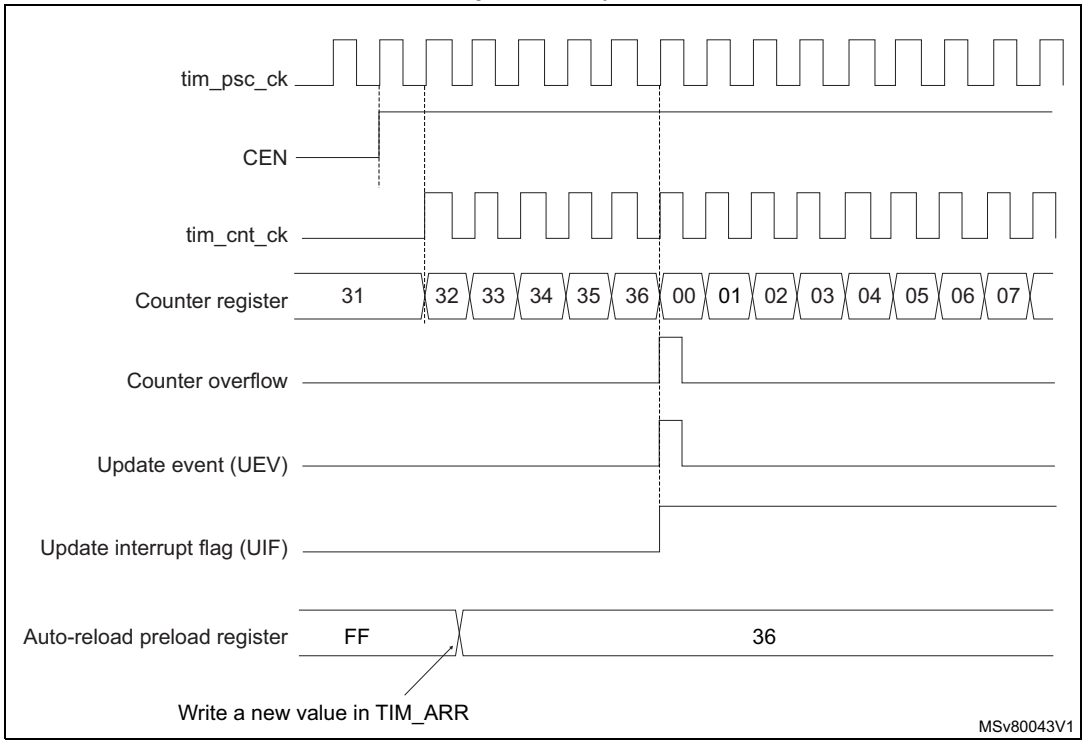
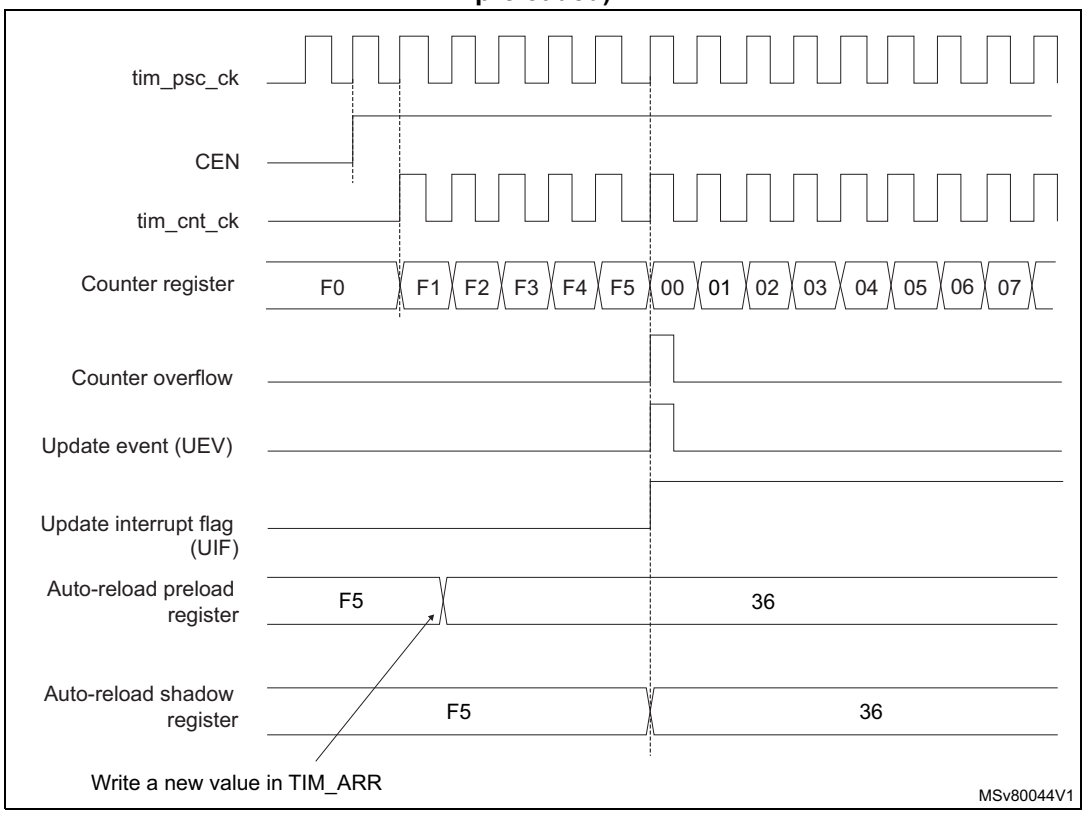


Figure 679. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



44.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1 (for TIM9/TIM12): external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (for TIM9/TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to *Section : Using one timer as prescaler for another timer* for more details.

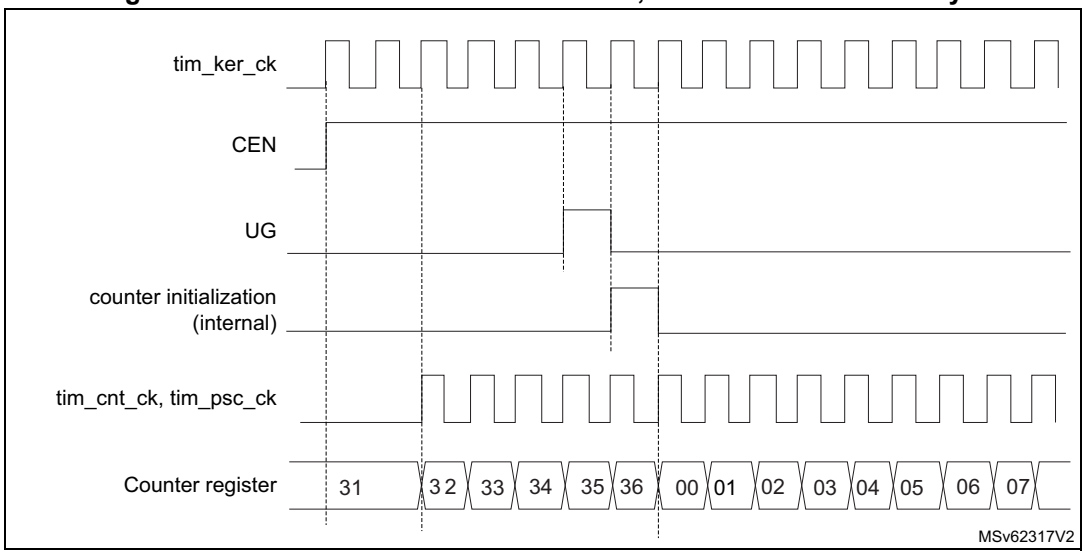
Internal clock source (tim_ker_ck)

The internal clock source is the default clock source for TIM10/TIM11/TIM13/TIM14.

For TIM9/TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS = 000). The CEN bit in the TIM_CR1 register and the UG bit in the TIM_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 680 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

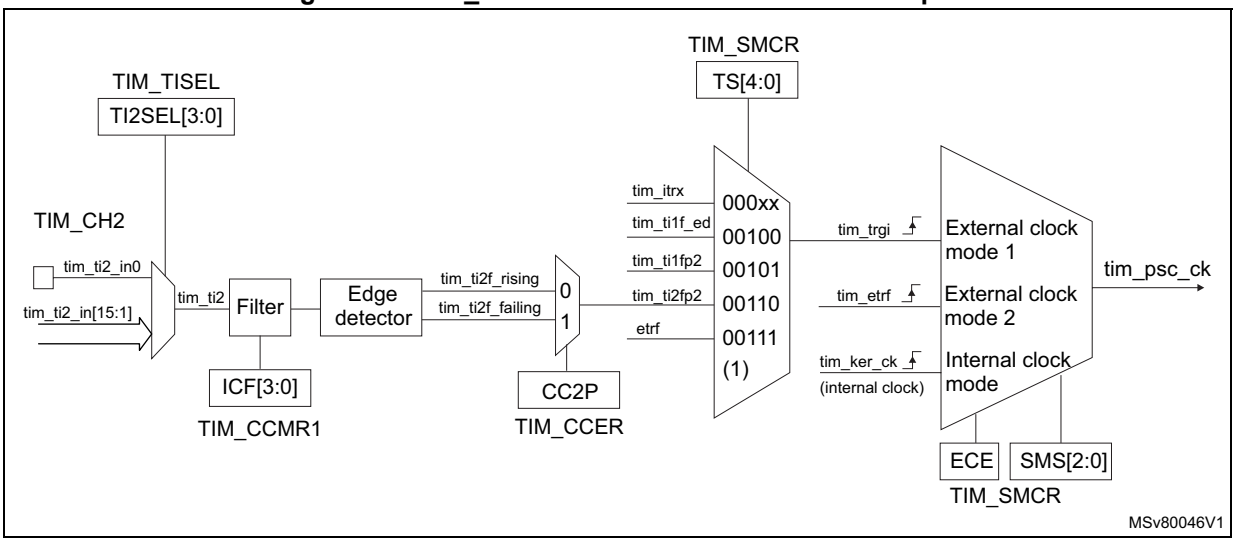
Figure 680. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1 (TIM9/TIM12 only)

This mode is selected when SMS = 111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 681. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

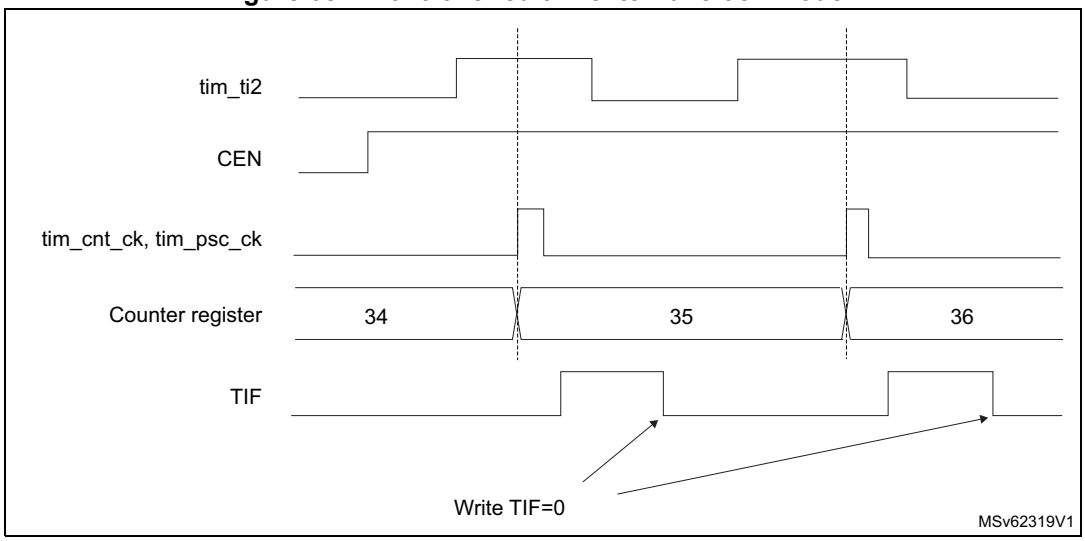
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select the rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 682. Control circuit in external clock mode 1



44.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler), and an output stage (with comparator and output control).

Figure 683, *Figure 684*, *Figure 685* and *Figure 686* give an overview of a capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 683. Capture/compare channel 1 input stage (TIM10/TIM11/TIM13/TIM14)

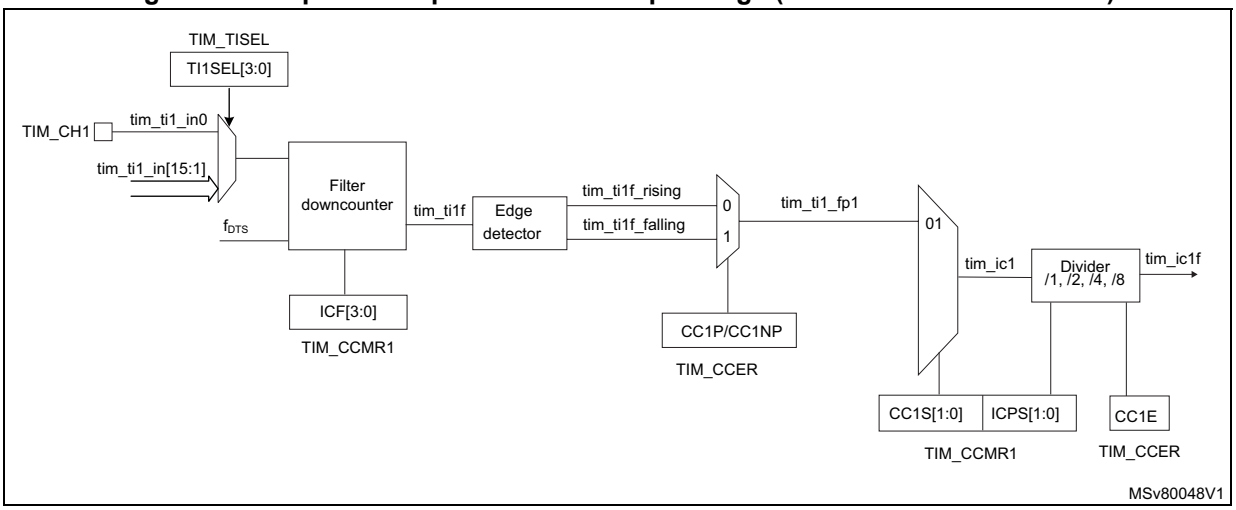
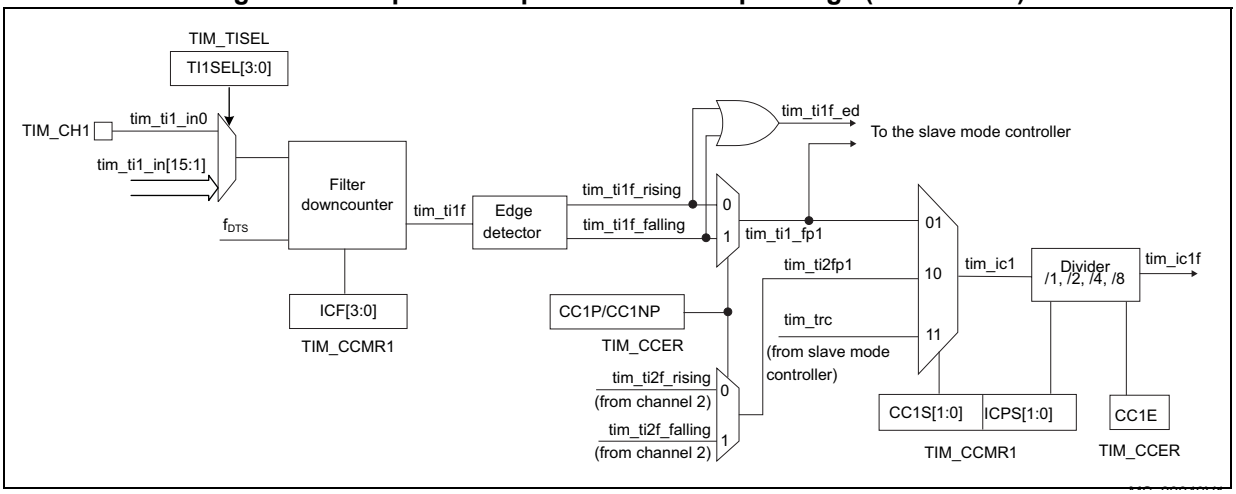


Figure 684. Capture/compare channel 1 input stage (TIM9/TIM12)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 685. Capture/compare channel 1 main circuit

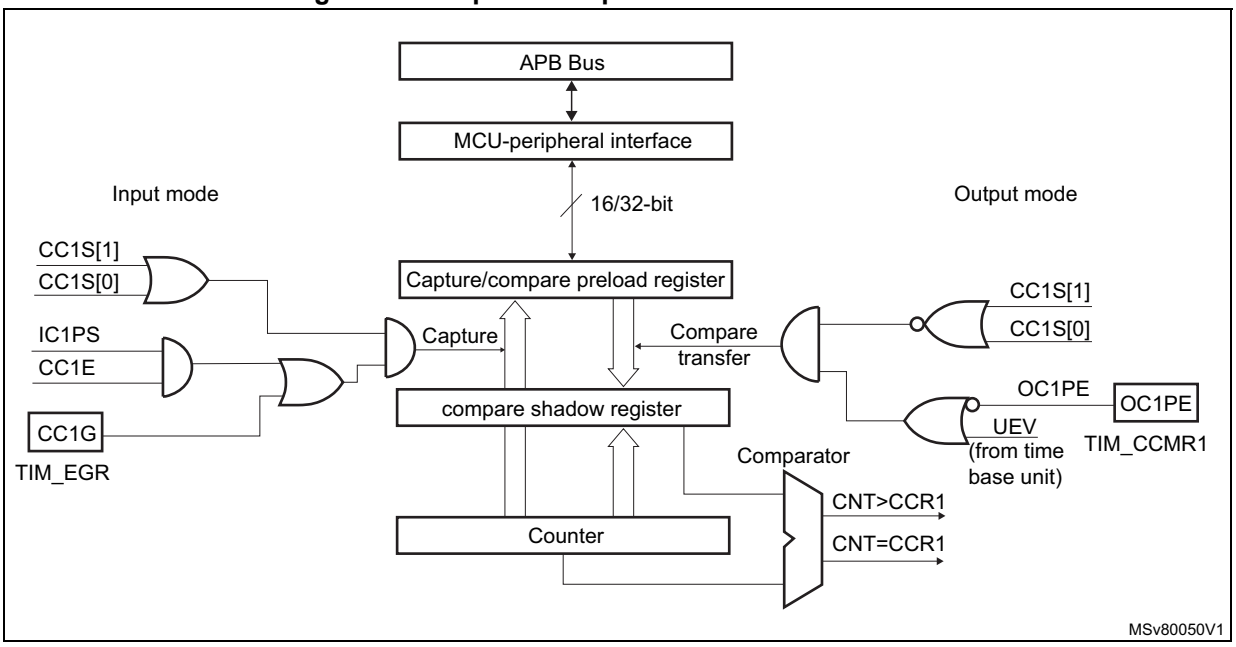
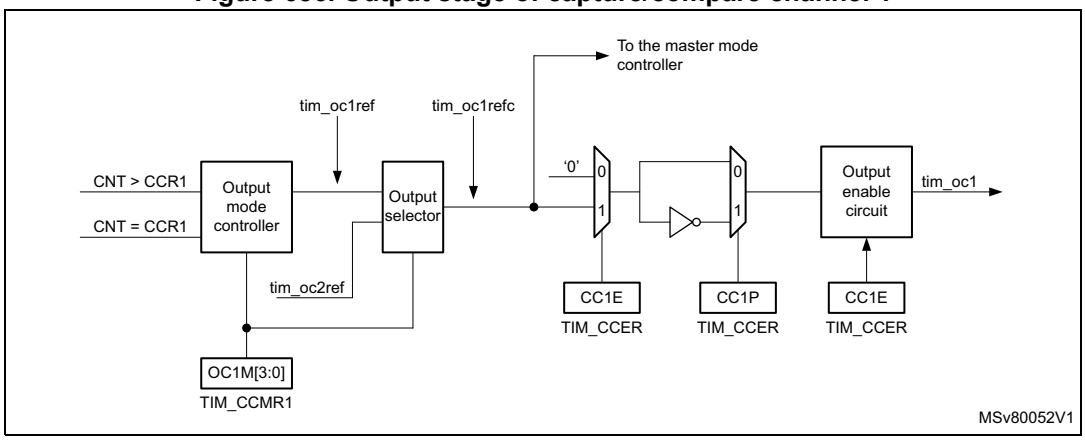


Figure 686. Output stage of capture/compare channel 1



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

44.4.7 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be

cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input mode, and the TIM_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIM_CCMRx register if the input is one of the tim_tix inputs). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by programming CC1P and CC1NP bits to 00 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wishes the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

44.4.8 PWM input mode (TIM9/TIM12 only)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges.

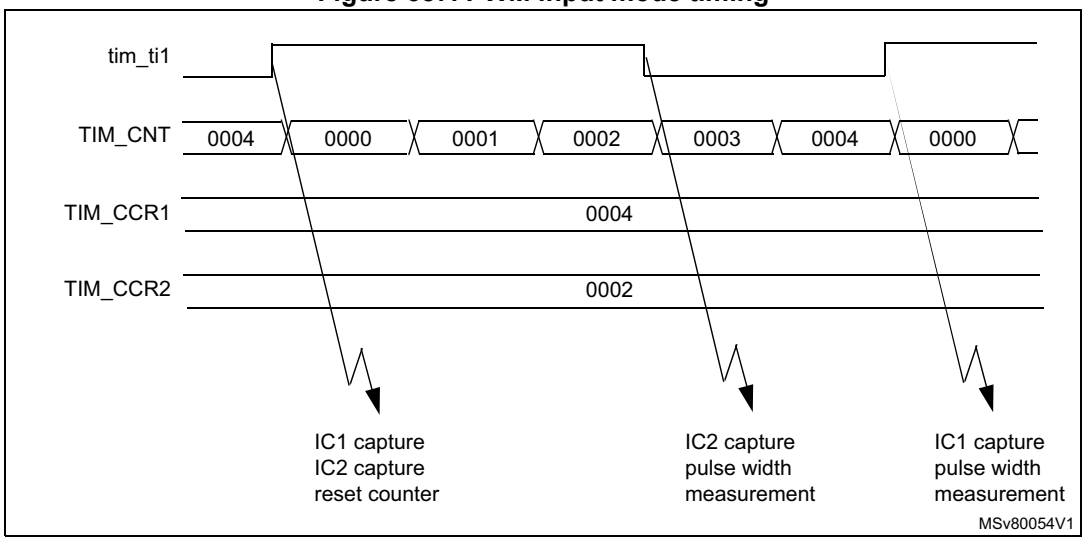
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIM_CCR1 register) and the duty cycle (in TIM_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): program the CC1P and CC1NP bits to 00 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): program the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 687. PWM input mode timing



44.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 0101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

44.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCxM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIM_DIER register).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

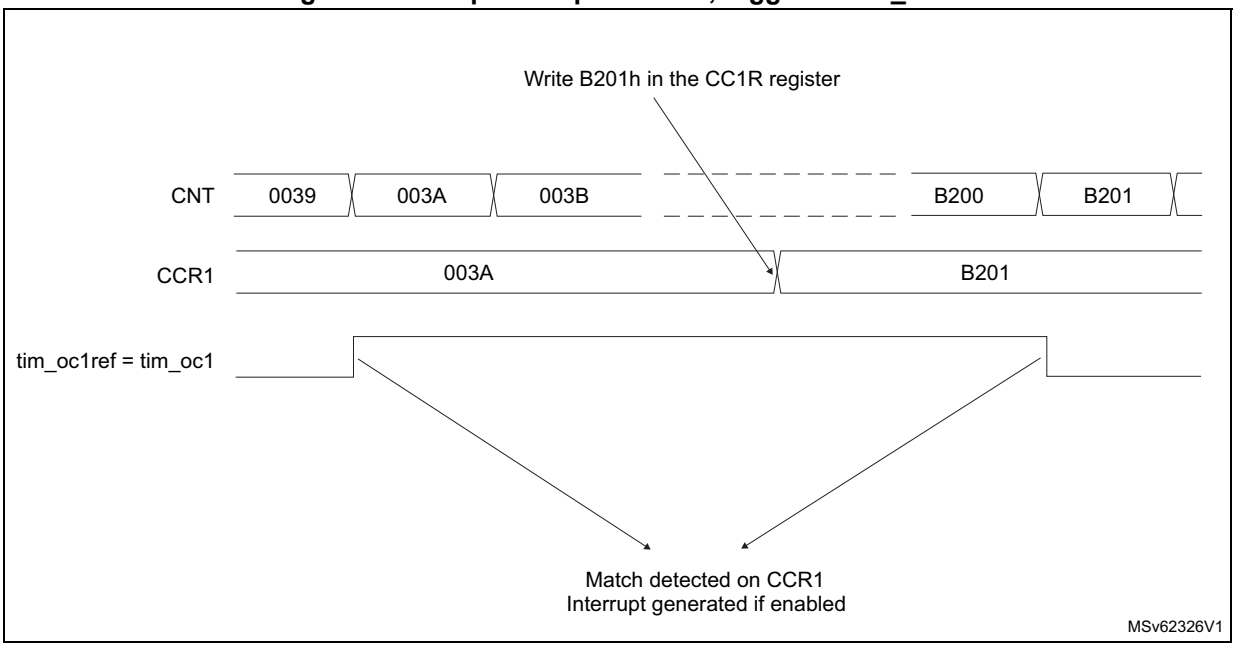
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 688*.

Figure 688. Output compare mode, toggle on tim_oc1.



44.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

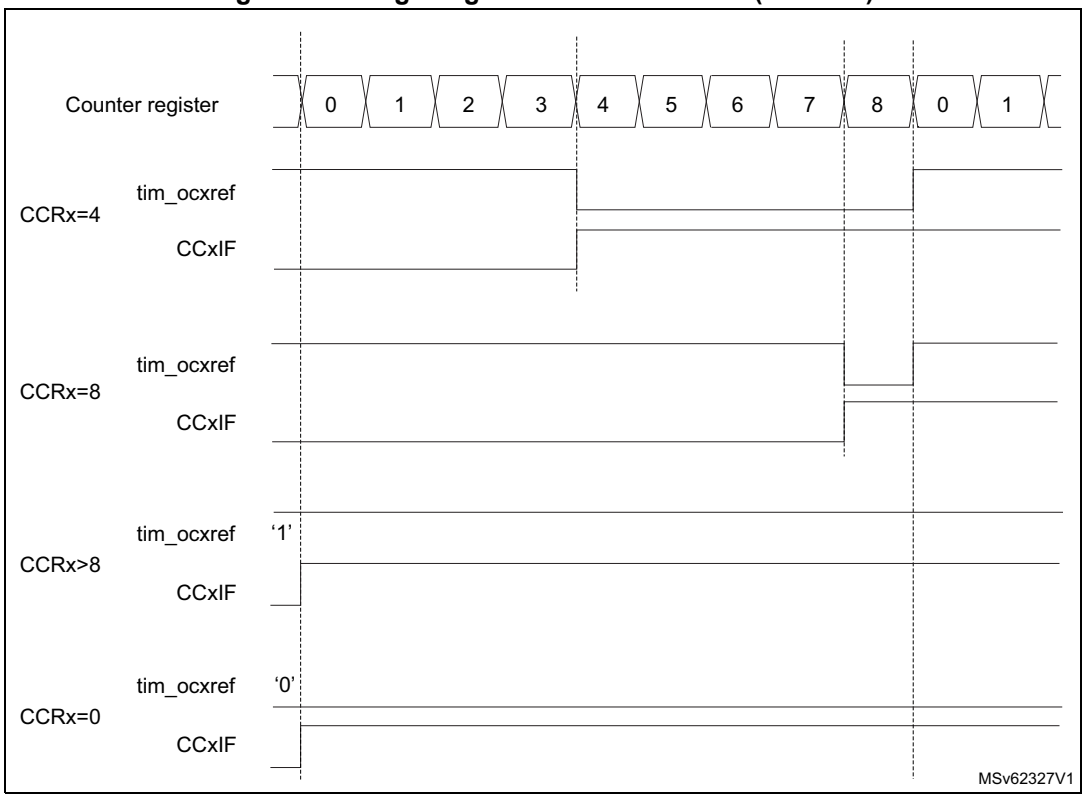
The tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. The tim_ocx output is enabled by the CCxE bit in the TIM_CCER register. Refer to the TIM_CCERx register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CNT ≤ TIM_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, the user considers PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 689* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 689. Edge-aligned PWM waveforms (ARR = 8)

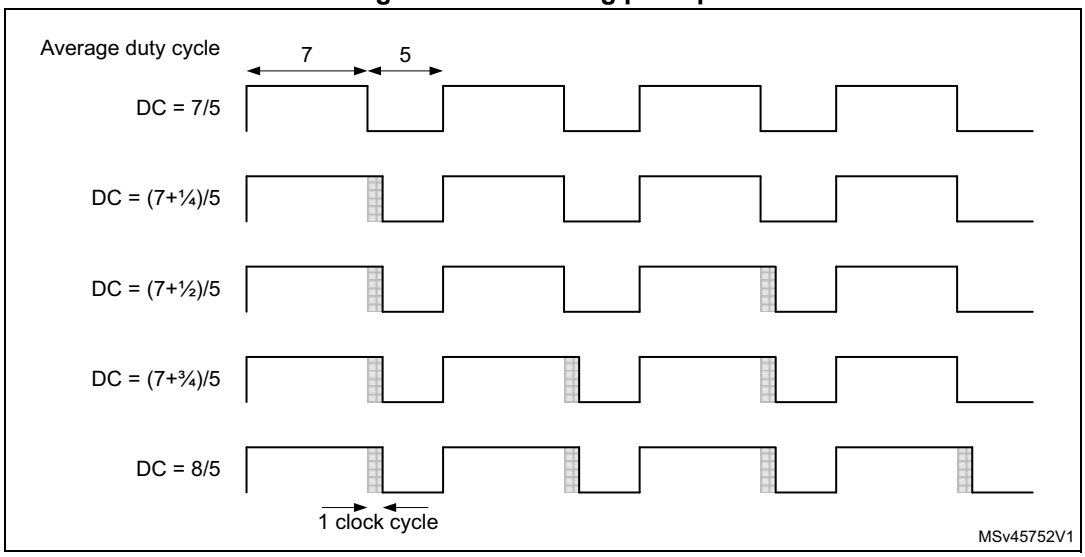


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 690* presents the dithering principle applied to four consecutive PWM cycles.

Figure 690. Dithering principle

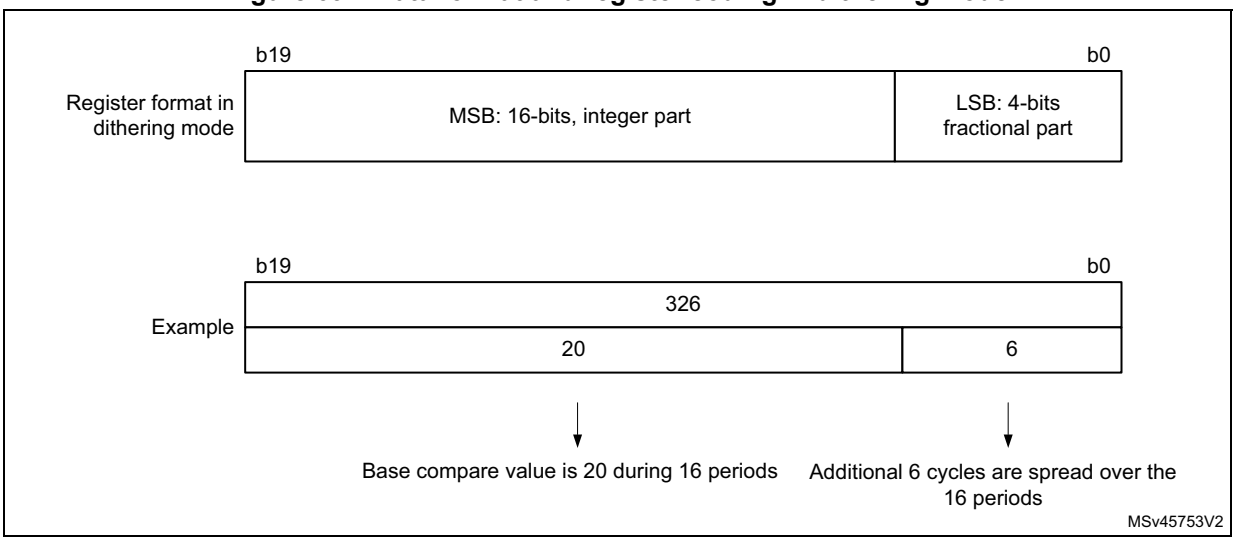


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 691* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 691. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

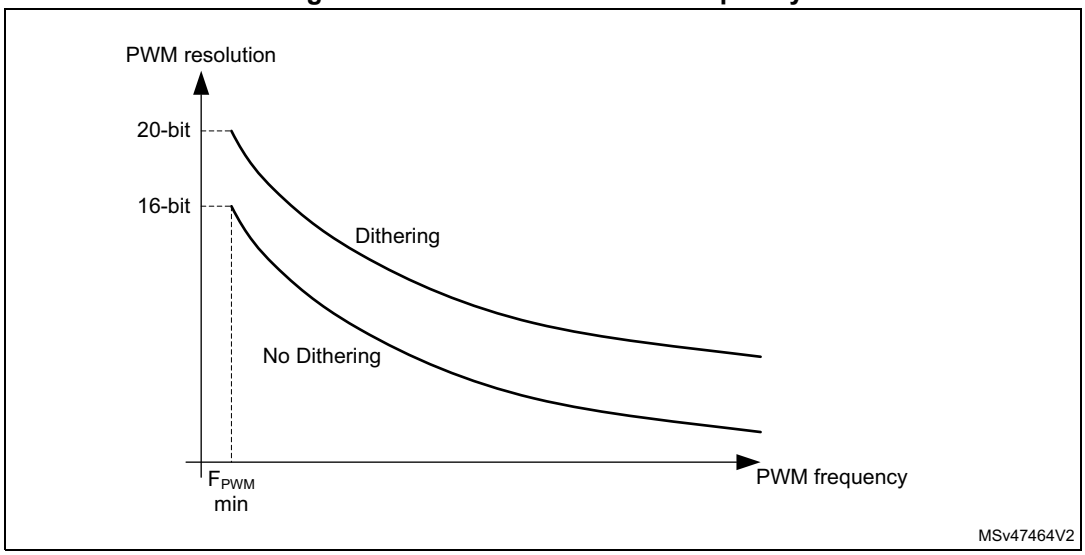
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

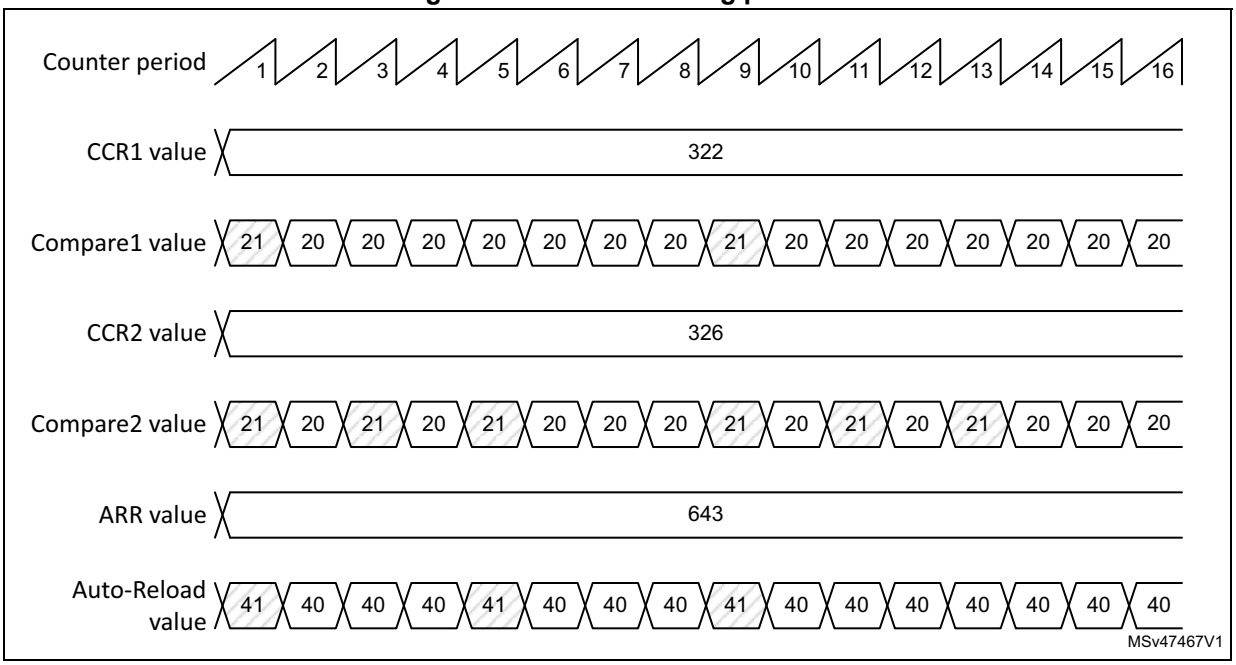
As shown on *Figure 692*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 692. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 693*.

Figure 693. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 507*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 507. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 507. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

44.4.12 Combined PWM mode (TIM9/TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIM_CCR1 and TIM_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

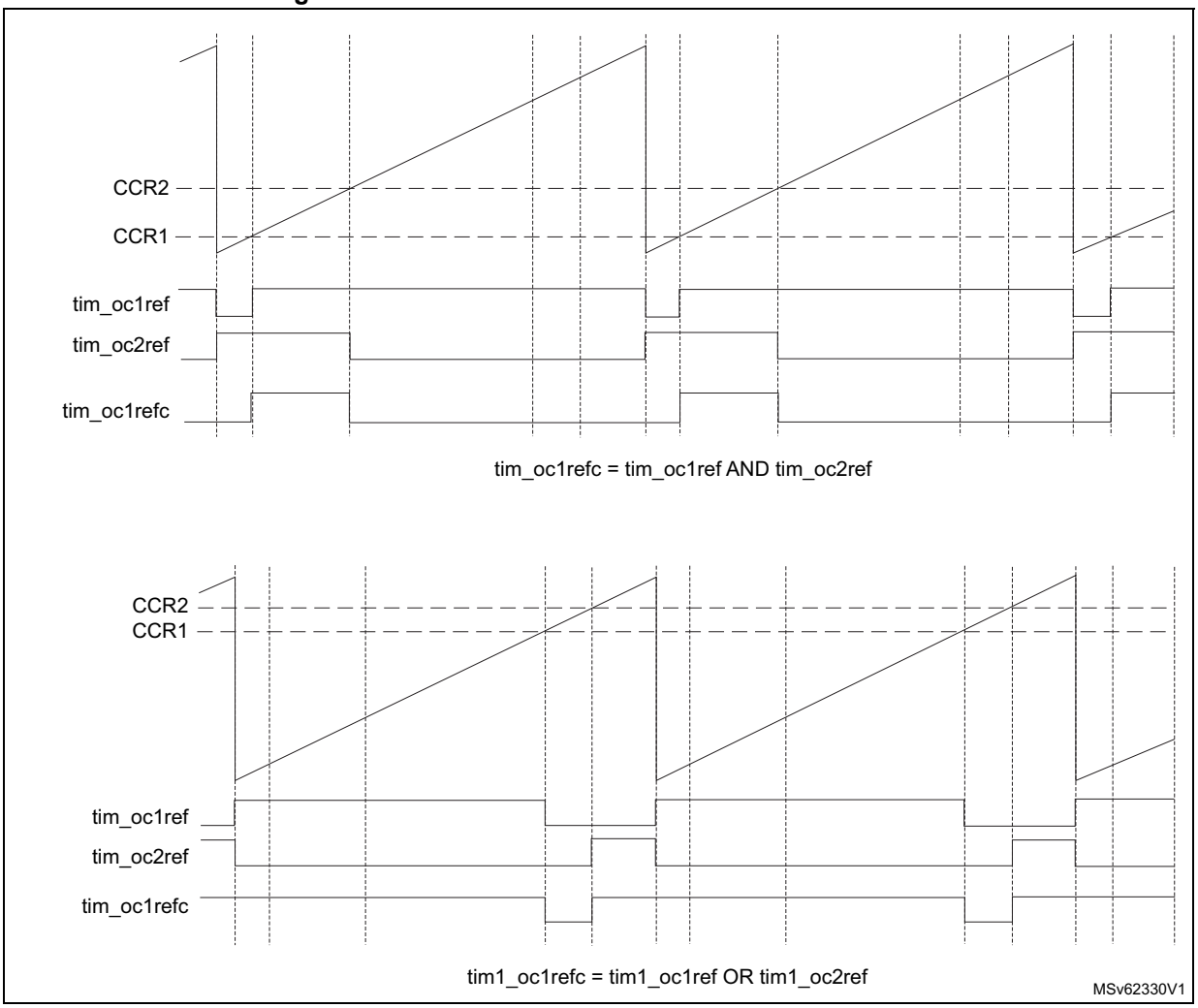
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 694 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 694. Combined PWM mode on channel 1 and 2

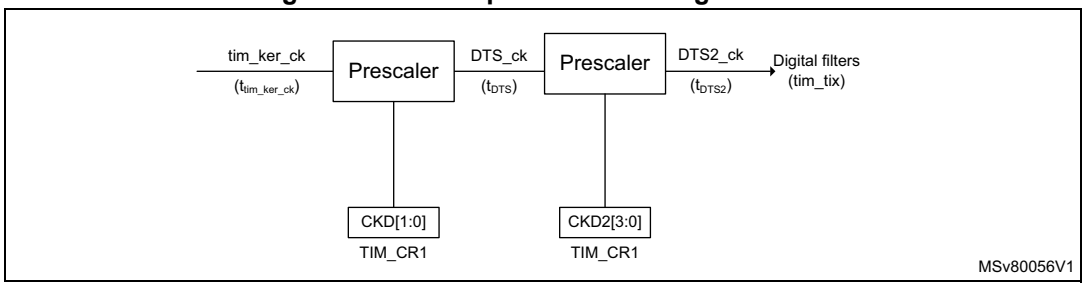


44.4.13 Clock dividers for digital filters

The digital filters (for tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the DTS_ck clock period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 695. Clock prescalers for digital filters



44.4.14 One-pulse mode

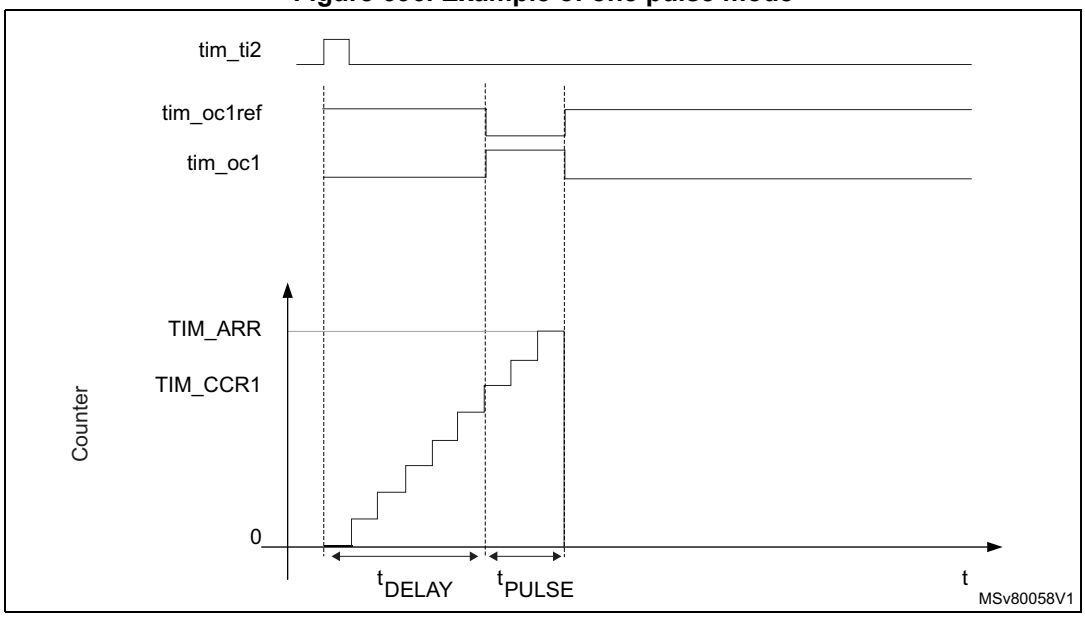
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 696. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Assuming the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 0111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIM_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

44.4.15 Retriggerable one pulse mode (TIM9/TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one pulse mode described in *Section 44.4.14: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

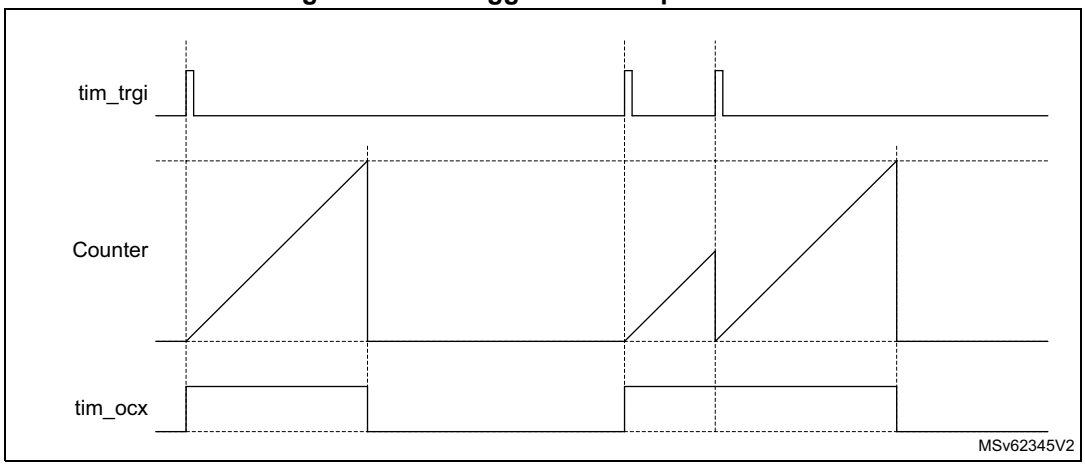
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 697. Retriggerable one pulse mode



44.4.16 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

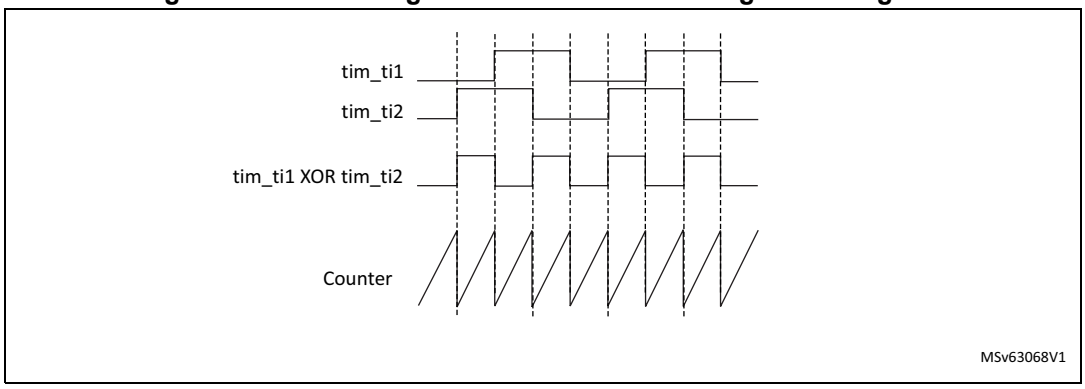
There is no latency between the assertions of the UIF and UIFCPY flags.

44.4.17 Timer input XOR function

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 698*.

Figure 698. Measuring time interval between edges on 2 signals



The XORPS control bit selects the position of the XOR gate, as shown in *Figure 699* and *Figure 700*, when the TI1S bit is set.

Figure 699. XOR gate connection with TI1S = 1 and XORPS = 0

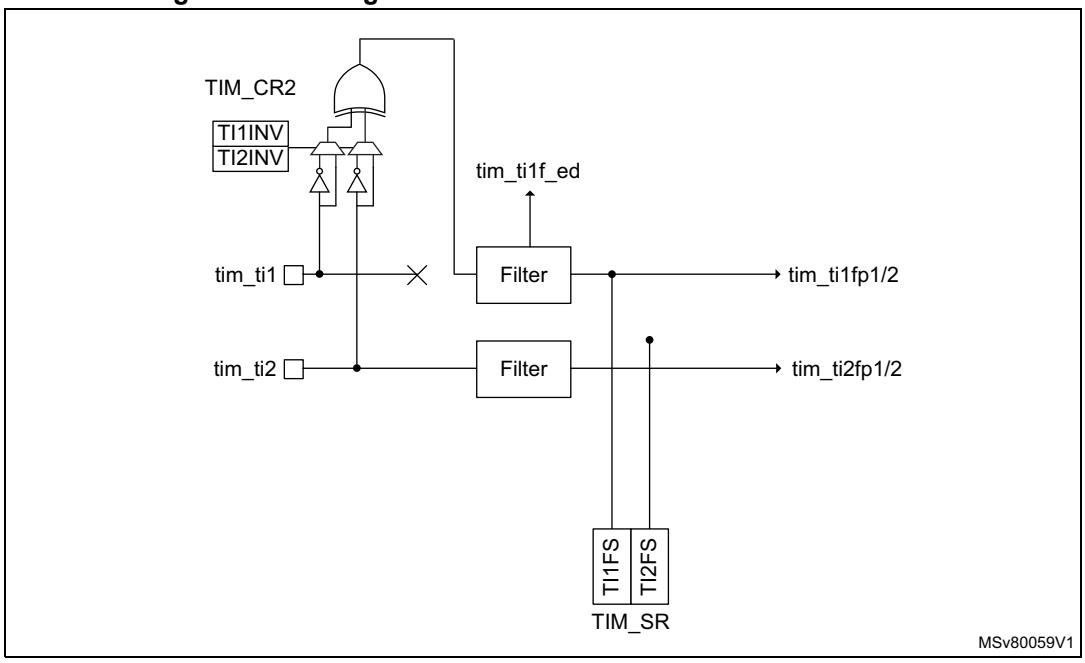
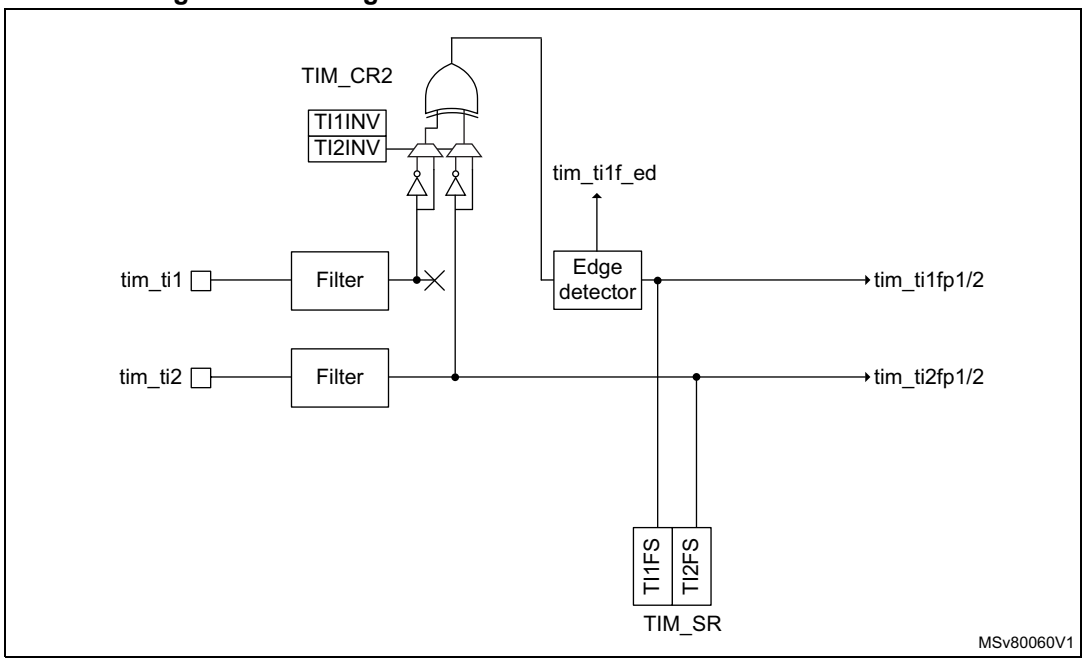


Figure 700. XOR gate connection with TI1S = 1 and XORPS = 1



Two control bits TI1INV and TI2INV in the TIM_CR2 register can invert the tim_in1 and tim_in2 signals prior entering the XOR gate.

The status of the tim_in1 and tim_in2 signals can be software polled using the TI1FS and TI2FS bits in the TIM_SR register.

44.4.18 TIM9/TIM12 external trigger synchronization

The TIM9/TIM12 timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and Gated + reset mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

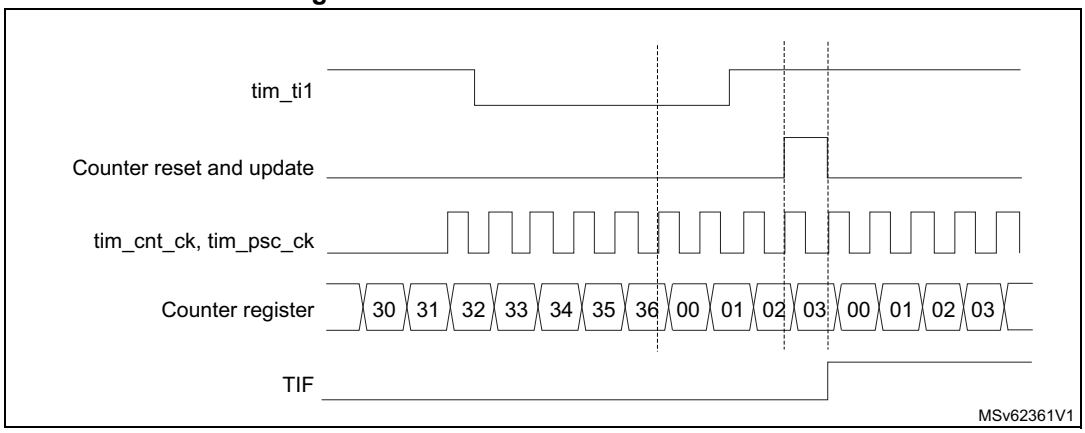
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Program CC1P and CC1NP to 00 in TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 701. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

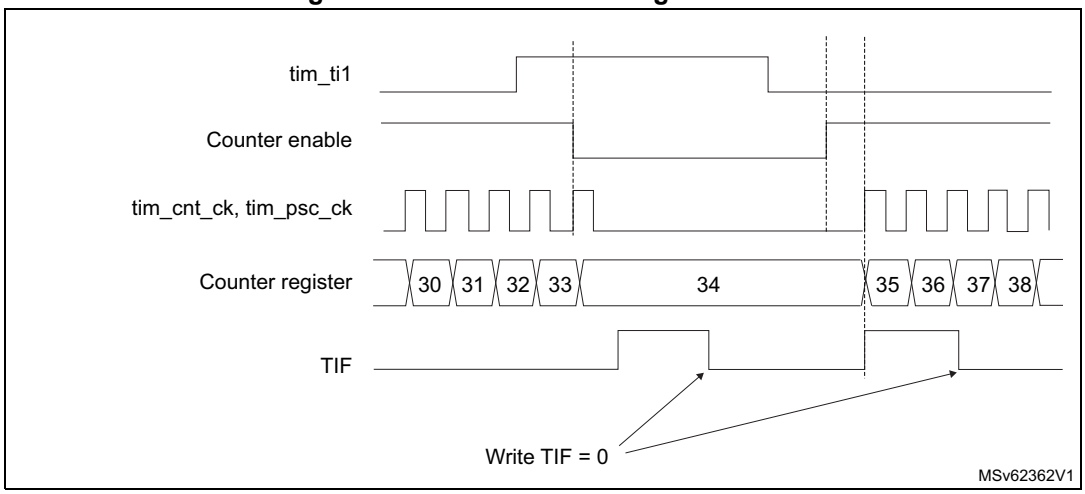
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Program CC1P = 1 and CC1NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 702. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

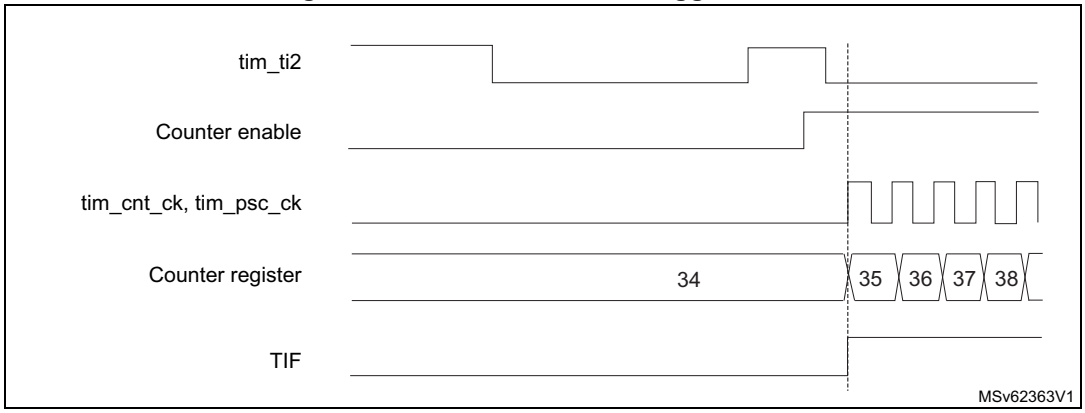
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Program CC2P = 1 and CC2NP = 0 in TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 703. Control circuit in trigger mode



44.4.19 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

44.4.20 Slave mode – combined reset + gated mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

44.4.21 Timer synchronization (TIM9/TIM12 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 42.4.27: Timer synchronization* for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

44.4.22 Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

44.4.23 ADC triggers (TIM9/TIM12 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

44.4.24 ADC synchronization (TIM9/TIM12 only)

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CCRx} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

44.4.25 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to the debug section.

44.5 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 low-power modes

Table 508. Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

44.6 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 interrupts

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 can generate multiple interrupts, as shown in *Table 509*.

Table 509. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
TIM	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No

1. Available for TIM9/TIM12 only.

44.7 TIM9/TIM12 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

44.7.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) and the digital filters (tim_tix).

0000: No Division
0001: $t_{DTS2} = 4 \times t_{DTS}$
0010: $t_{DTS2} = 16 \times t_{DTS}$
0011: $t_{DTS2} = 64 \times t_{DTS}$
0100: $t_{DTS2} = 256 \times t_{DTS}$
0101: $t_{DTS2} = 1024 \times t_{DTS}$
0110: $t_{DTS2} = 4096 \times t_{DTS}$
0111: $t_{DTS2} = 16384 \times t_{DTS}$
1000: $t_{DTS2} = 65536 \times t_{DTS}$
1001: $t_{DTS2} = 262144 \times t_{DTS}$
Others: Reserved, do not program this value.

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: $t_{DTS} = 8 \times t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered.
1: TIM_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt if enabled. These events can be:
– Counter overflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update event (UEV) generation.
0: UEV enabled. An UEV is generated by one of the following events:
– Counter overflow
– Setting the UG bit
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
CEN is cleared automatically in one-pulse mode, when an update event occurs.
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

44.7.2 TIM control register 2 (TIM_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TI2INV	TI1INV	AD SYNC	XORPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input
0: tim_ti2 non-inverted
1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input
0: tim_ti1 non-inverted
1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 44.4.24* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection

This bit defines the position of the XOR gate with respect to input filters.

0: XOR gate placed before TI1 filter

1: XOR gate placed after TI1 and TI2 filters

Bits 26:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

This bit enables the insertion of an XOR gate to combine tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs into tim_ti1.

The position of the XOR gate, with respect to input filters, is defined with the XORPS bit.

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: Reset - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

010: Update - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: Compare - tim_oc1refc signal is used as trigger output (tim_trgo).

101: Compare - tim_oc2refc signal is used as trigger output (tim_trgo).

Bits 3:0 Reserved, must be kept at reset value.

44.7.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/Slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful in order to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)

See *Section 44.4.2: TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals* for more details on the meaning of tim_itrx for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see input control register and control register description.

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Other codes: reserved.

Note: The gated mode (including gated + reset mode) must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signals must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

44.7.4 TIM interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled.

1: Trigger interrupt enabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

0: CC2 interrupt disabled.

1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

44.7.5 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r				rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update interrupt overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No update interrupt overrun
1: Update interrupt overrun

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.
If channel CC1 is configured as output: this flag is set when he content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow and if UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 44.7.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

44.7.6 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIM_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

44.7.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample the tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

44.7.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas the active level of tim_oc1 depends on the CC1P.

0000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The tim_oc1ref signal is forced high when the TIM_CNT counter matches the capture/compare register 1 (TIM_CCR1).

0010: Set channel 1 to inactive level on match. The tim_oc1ref signal is forced low when the TIM_CNT counter matches the capture/compare register 1 (TIM_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1

0100: Force inactive level - tim_oc1ref is forced low

0101: Force active level - tim_oc1ref is forced high

0110: PWM mode 1 - channel 1 is active as long as TIM_CNT<TIM_CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else it is active

1000: Retrigerrable OPM mode 1 - The channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retrigerrable OPM mode 2 - The channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00:CC1 channel is configured as output
01:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11:CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

44.7.9 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
CC1 channel configured as output: CC1NP must be kept cleared
CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 and tim_ti2fp1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_tixfp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_tixfp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_tixfp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / tim_oc1 is not active

1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 510. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The states of the external I/O pins connected to the standard tim_ocx *channels depend on the state of the* tim_ocx *channel and on the GPIO registers.*

44.7.10 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

44.7.11 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

44.7.12 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 44.4.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

44.7.13 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

44.7.14 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

44.7.15 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIM_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

0100: tim_ti2_in15

Refer to *Section 44.4.2: TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIM_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 44.4.2: TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals* for interconnects list.

44.7.16 TIM9/TIM12 register map

TIM9/TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 511. TIM9/TIM12 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0				0	0	0	0
0x004	**TIM_CR2**	Res.	TI2INV	TI1INV	ADSYNC	XORPS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value		0	0	0	0																				0	0	0	0				
0x008	**TIM_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x00C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value								0																		0				0	0	0
0x010	**TIM_SR**	Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value			0	0				0														0	0			0				0	0	0
0x014	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x018	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x024	**TIM_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 511. TIM9/TIM12 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIM_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C to 0x058	Reserved	Res.																															
0x05C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

44.8 TIM10/TIM11/TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

44.8.1 TIM control register 1 (TIM_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) and the digital filters (tim_tix).

0000: No Division
0001: $t_{DTS2} = 4 \times t_{DTS}$
0010: $t_{DTS2} = 16 \times t_{DTS}$
0011: $t_{DTS2} = 64 \times t_{DTS}$
0100: $t_{DTS2} = 256 \times t_{DTS}$
0101: $t_{DTS2} = 1024 \times t_{DTS}$
0110: $t_{DTS2} = 4096 \times t_{DTS}$
0111: $t_{DTS2} = 16384 \times t_{DTS}$
1000: $t_{DTS2} = 65536 \times t_{DTS}$
1001: $t_{DTS2} = 262144 \times t_{DTS}$
Others: Reserved, do not program this value.

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: $t_{DTS} = 8 \times t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the update interrupt (UEV) sources.
0: Any of the following events generate an UEV if enabled:
– Counter overflow
– Setting the UG bit
1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.
0: UEV enabled. An UEV is generated by one of the following events:
– Counter overflow
– Setting the UG bit.
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

44.8.2 TIM interrupt enable register (TIM_DIER)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

44.8.3 TIM status register (TIM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			r				rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TI1FS**: tim_ti1f status

This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.

0: tim_ti1f signal is low

1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.

0: No Update Interrupt Overrun

1: Update Interrupt Overrun

Bits 23:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when he content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow and if UDIS = 0 in the TIM_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

44.8.4 TIM event generation register (TIM_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
														w	w

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

44.8.5 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

44.8.6 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.
0000: Frozen. The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.
0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).
0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).
0011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.
0100: Force inactive level - tim_oc1ref is forced low.
0101: Force active level - tim_oc1ref is forced high.
0110: PWM mode 1 - Channel 1 is active as long as TIM_CNT < TIM_CCR1 else inactive.
0111: PWM mode 2 - Channel 1 is inactive as long as TIM_CNT < TIM_CCR1 else active
Others: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: Reserved.

11: Reserved.

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

44.8.7 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.

CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define tim_ti1fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 for capture operations.
CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_ti1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_ti1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_ti1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 not inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 512. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output Disabled (tim_ocx = 0)
1	tim_ocx = tim_ocxref + Polarity

Note: The state of the external I/O pins connected to the standard tim_ocx channels depends on the tim_ocx channel state and the GPIO registers.

44.8.8 TIM counter (TIM_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIM_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

44.8.9 TIM prescaler (TIM_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

(including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

44.8.10 TIM autoreload register (TIM_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 44.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

44.8.11 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

44.8.12 TIM timer input selection register (TIM_TISEL)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIM_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 44.4.2: TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals* for interconnects list.

44.8.13 TIM10/TIM11/TIM13/TIM14 register map

TIM10/TIM11/TIM13/TIM14 registers are mapped as 16-bit addressable registers as described in the tables below:

Table 513. TIM10/TIM11/TIM13/TIM14 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0				0	0	0	0
0x004 to 0x008	Reserved	Res.																															
0x00C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
	Reset value																															0	0
0x010	**TIM_SR**	Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
	Reset value				0				0															0								0	0
0x014	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
	Reset value																															0	0
0x018	**TIM_CCMR1** Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIM_CCMR1 Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
	Reset value																													0		0	0

Table 513. TIM10/TIM11/TIM13/TIM14 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	**TIM_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x058	Reserved	Res.																															
0x05C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.3* for the register boundary addresses.

45 General purpose timers (TIM15/TIM16/TIM17)

45.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 45.4.33: Timer synchronization (TIM15 only)*.

45.2 TIM15 main features

TIM15 includes the following features:

- 16-bit up, down, up/down autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
- Incremental encoders and quadrature encoders and hall sensors support
- ADC synchronization for jitter-free sampling points

45.3 TIM16/TIM17 main features

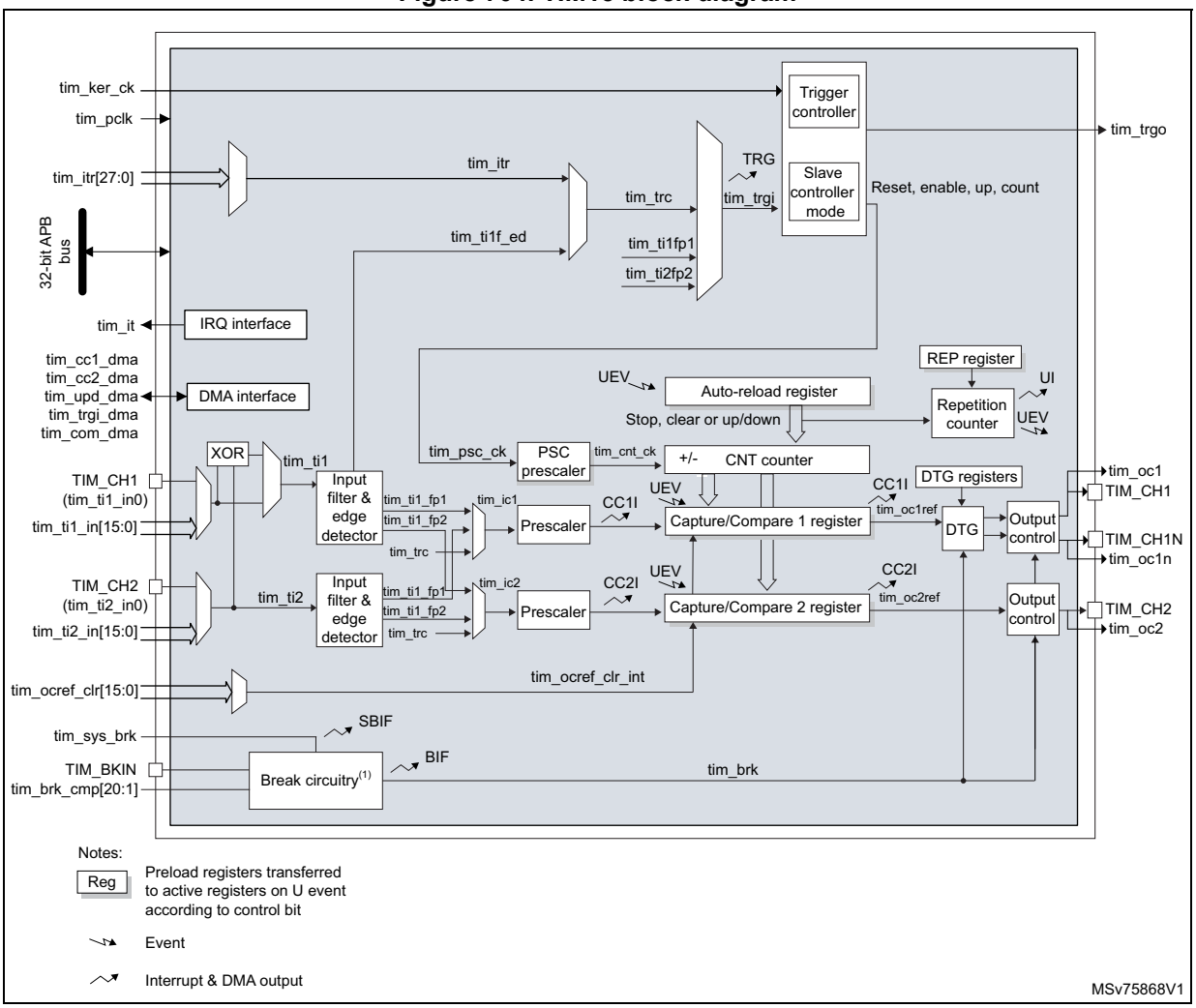
The TIM16/TIM17 timers include the following features:

- 16-bit up, down, up/down autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

45.4 TIM15/TIM16/TIM17 functional description

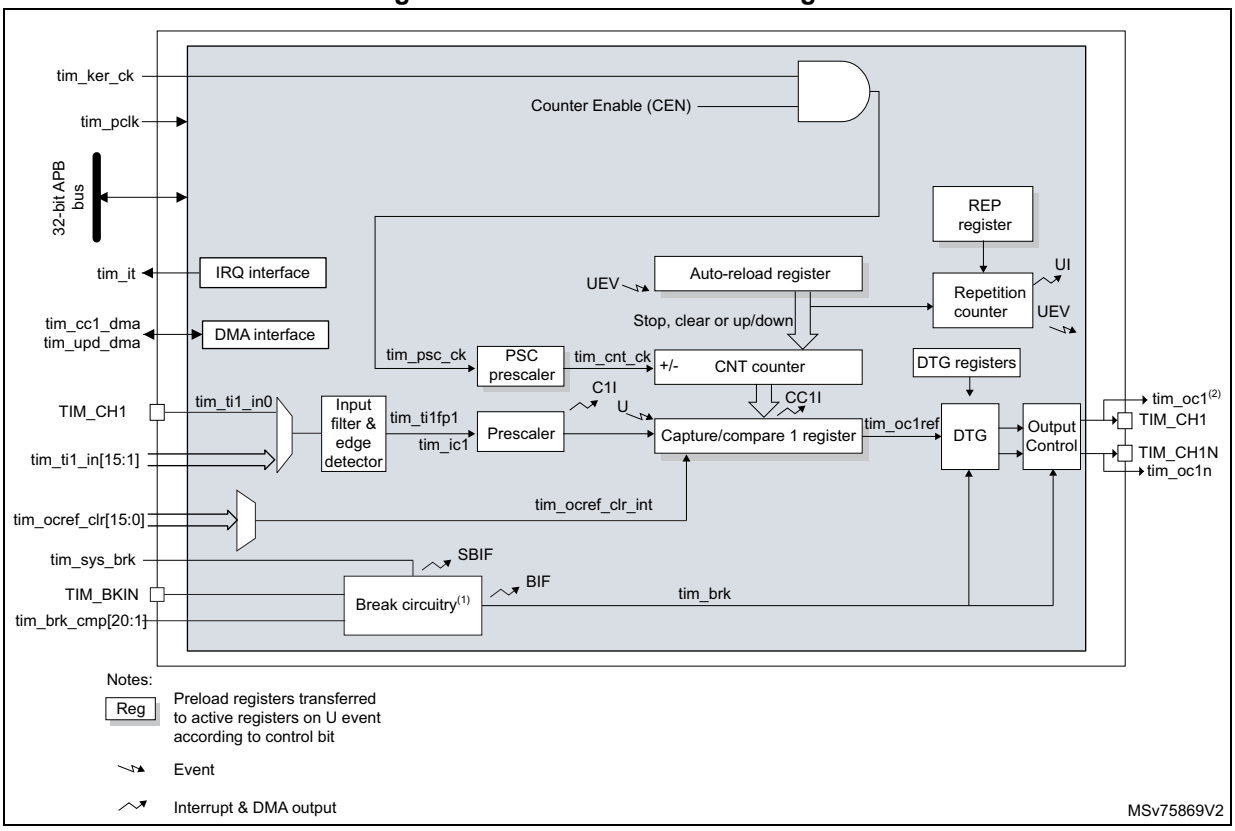
45.4.1 Block diagram

Figure 704. TIM15 block diagram



1. Refer to *Section 45.4.17: Using the break function* for details.

Figure 705. TIM16/TIM17 block diagram



1. Refer to *Section 45.4.17: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 45.4.34: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

45.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 514 and *Table 515* in this section summarize the TIM inputs and outputs.

Table 514. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

1. Available for TIM15 only.

Table 515. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_itr[27:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocx (x = 1 to 2)(1)	Output	Internal output compare signals of channels 1 to 2. These outputs can trigger or synchronize other internal peripherals.
tim_oc1n	Output	Internal timer complementary output derived from tim_oc1 internal output, with the option to insert dead time.
tim_ocref_clr[15:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[20:1]	Input	Break input for internal signals
tim_sys_brk[5:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_cc2_dma(1)	Output	Timer capture / compare 2 dma request
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

1. Available for TIM15 only.

Table 516 and *Table 517* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 516. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	comp1_out	comp6_out	comp5_out
tim_ti1_in2	RCC_MCO1	RCC_LSI	RCC_HSE
tim_ti1_in3	RCC_MCO2	RCC_LSE	RCC_MCO1
tim_ti1_in4	hsi_cal_ck	RTC_WKUP_IT	-
tim_ti1_in5	-	-	-
tim_ti1_in6	comp4_out	-	-
tim_ti1_in7	comp5_out	-	-
tim_ti1_in8	comp7_out	-	-
tim_ti1_in9	play1_out4	play1_out5	play1_out9
tim_ti1_in10	RCC_LSE	-	-
tim_ti1_in[15:11]	Reserved		

Table 517. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM15
tim_ti2_in0	TIM15_CH2
tim_ti2_in1	comp2_out
tim_ti2_in2	comp3_out
tim_ti2_in3	comp6_out
tim_ti2_in4	comp8_out
tim_ti2_in[8:5]	Reserved
tim_ti2_in9	play1_out5(1)
tim_ti2_in[15:10]	Reserved

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability.

Table 518 lists the internal sources connected to the tim_itr input multiplexer.

Table 518. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim1_trgo2
tim_itr2	tim2_trgo
tim_itr3	tim3_trgo(1)
tim_itr4	tim4_trgo(1)
tim_itr5	tim5_trgo(1)
tim_itr6	tim8_trgo
tim_itr7	tim8_trgo2
tim_itr8	tim9_trgo
tim_itr9	tim10_oc1
tim_itr10	tim11_oc1
tim_itr11	tim12_trgo
tim_itr12	tim13_oc1
tim_itr13	tim14_oc1
tim_itr14	-
tim_itr15	tim16_oc1
tim_itr16	tim17_oc1
tim_itr17	tim20_trgo(1)
tim_itr18	tim20_trgo2(1)
tim_itr19	-
tim_itr20	-
tim_itr21	hrtim1_out_sync2
tim_itr[27:22]	Reserved

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability.

Table 519 and *Table 520* list the sources connected to the tim_brk input.

Table 519. Timer break interconnect

tim_brk inputs	TIM15	TIM16	TIM17
tim_brk_cmp0	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out	comp1_out
tim_brk_cmp2	comp2_out	comp2_out	comp2_out
tim_brk_cmp3	comp3_out	comp3_out	comp3_out

Table 519. Timer break interconnect (continued)

tim_brk inputs	TIM15	TIM16	TIM17
tim_brk_cmp4	comp4_out	comp4_out	comp4_out
tim_brk_cmp5	comp5_out	comp5_out	comp5_out
tim_brk_cmp6	comp6_out	comp6_out	comp6_out
tim_brk_cmp7	comp7_out	comp7_out	comp7_out
tim_brk_cmp8	comp8_out	comp8_out	comp8_out
tim_brk_cmp9	TIM1_BKIN2	TIM1_BKIN2	TIM1_BKIN2
tim_brk_cmp10	TIM8_BKIN2	TIM8_BKIN2	TIM8_BKIN2
tim_brk_cmp11	TIM20_BKIN2(1)	TIM20_BKIN2(1)	TIM20_BKIN2(1)
tim_brk_cmp12	-	-	-
tim_brk_cmp13	play1_out4(1)	play1_out4(1)	play1_out4(1)
tim_brk_cmp14	play1_out5(1)	play1_out7(1)	play1_out12(1)
tim_brk_cmp15	TIM1_BKIN1	TIM1_BKIN1	TIM1_BKIN1
tim_brk_cmp16	TIM8_BKIN1	TIM8_BKIN1	TIM8_BKIN1
tim_brk_cmp17	TIM20_BKIN1(1)	TIM20_BKIN1(1)	TIM20_BKIN1(1)
tim_brk_cmp18	-	TIM15_BKIN1	TIM15_BKIN1
tim_brk_cmp19	TIM16_BKIN1	-	TIM16_BKIN1
tim_brk_cmp20	TIM17_BKIN1	TIM17_BKIN1	-

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability.

Table 520. System break interconnect

tim_sys_brk inputs	TIM15 / TIM16 / TIM17	Enable bit in SBS_CFGR2 register
tim_sys_brk0	Cortex-M33 LOCKUP	CLL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	SRAM double ECC error	SEL
tim_sys_brk3	Flash double ECC error	ECCL
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)
tim_sys_brk5	Cortex-M33 data cache double ECC error	DCECCL

Table 521 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 521. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment		
	TIM15	TIM16	TIM17
tim_ocref_clr0	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out
tim_ocref_clr2	comp3_out	comp3_out	comp3_out
tim_ocref_clr3	comp4_out	comp4_out	comp4_out
tim_ocref_clr4	comp5_out	comp5_out	comp5_out
tim_ocref_clr5	comp6_out	comp6_out	comp6_out
tim_ocref_clr6	comp7_out	comp7_out	comp7_out
tim_ocref_clr7	comp8_out	comp8_out	comp8_out
tim_ocref_clr[11:8]	Reserved		
tim_ocref_clr12	play1_out7(1)	play1_out9(1)	play1_out11(1)
tim_ocref_clr13	play1_out8(1)	play1_out10(1)	play1_out13(1)
tim_ocref_clr[15:14]	Reserved		

1. Not available on all STM32G56x/55x/54x/53x devices. Refer to the device datasheet for availability.

45.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter can count up, down or both up and down.The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIM_CNT)
- Prescaler register (TIM_PSC)
- Autoreload register (TIM_ARR)
- Repetition counter register (TIM_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIM_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIM_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

Note: An update overrun status bit (UIOVRF in TIM_SR register) indicates that an update request occurred while an update interrupt is pending (update interrupt flag set). This bit is useful when debugging closed loop applications, to indicate that:

- the update interrupt service routine is too long with respect to the interval between consecutive update events

- the update interrupt service routine was served too late with respect to the subsequent update event

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIM_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIM_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIM_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 706 and *Figure 707* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 706. Counter timing diagram with prescaler division change from 1 to 2

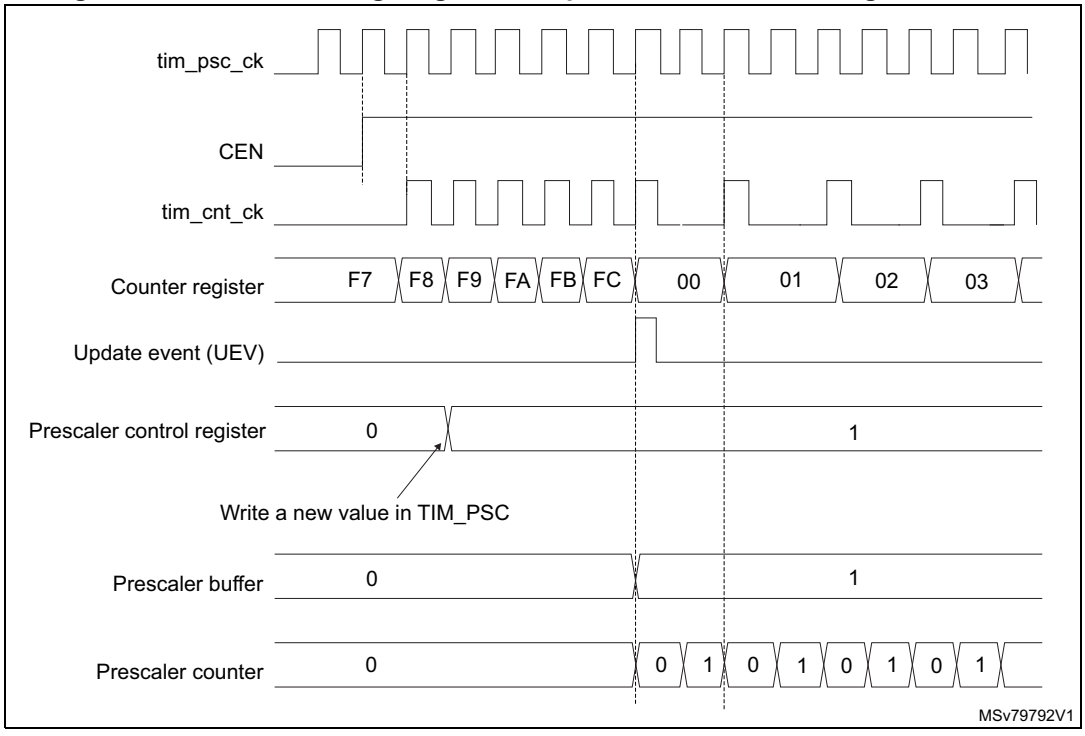
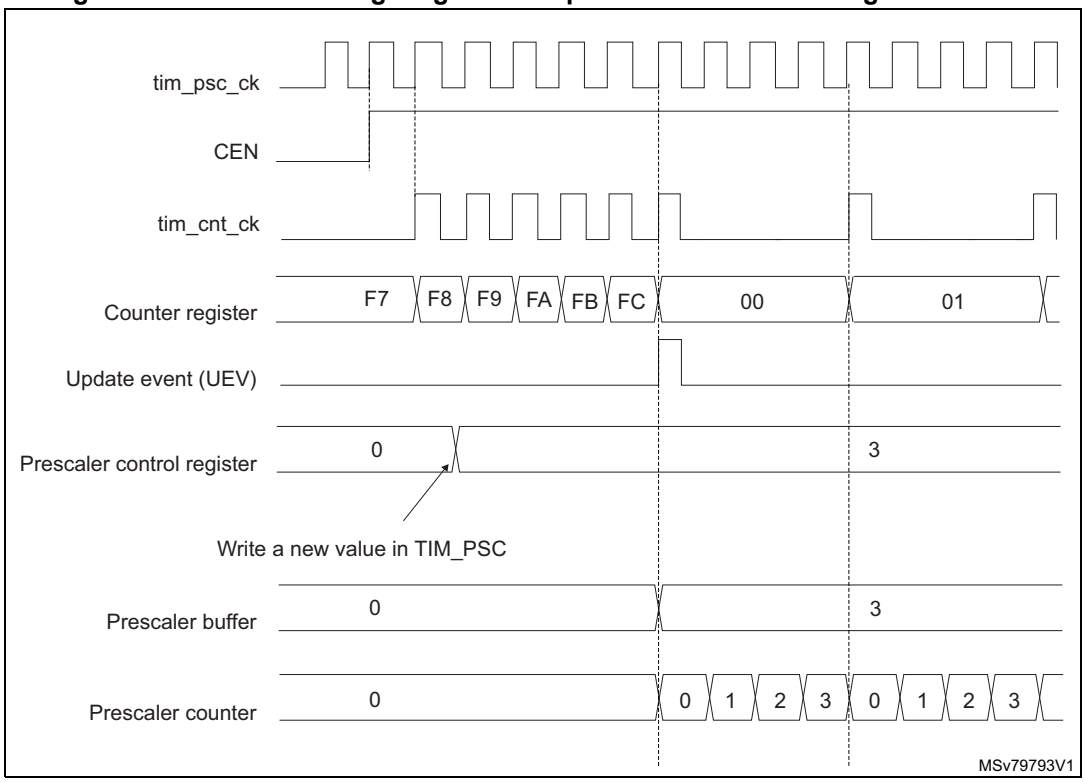


Figure 707. Counter timing diagram with prescaler division change from 1 to 4



45.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register,
- The autoreload shadow register is updated with the preload value (TIM_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 708. Counter timing diagram, internal clock divided by 1

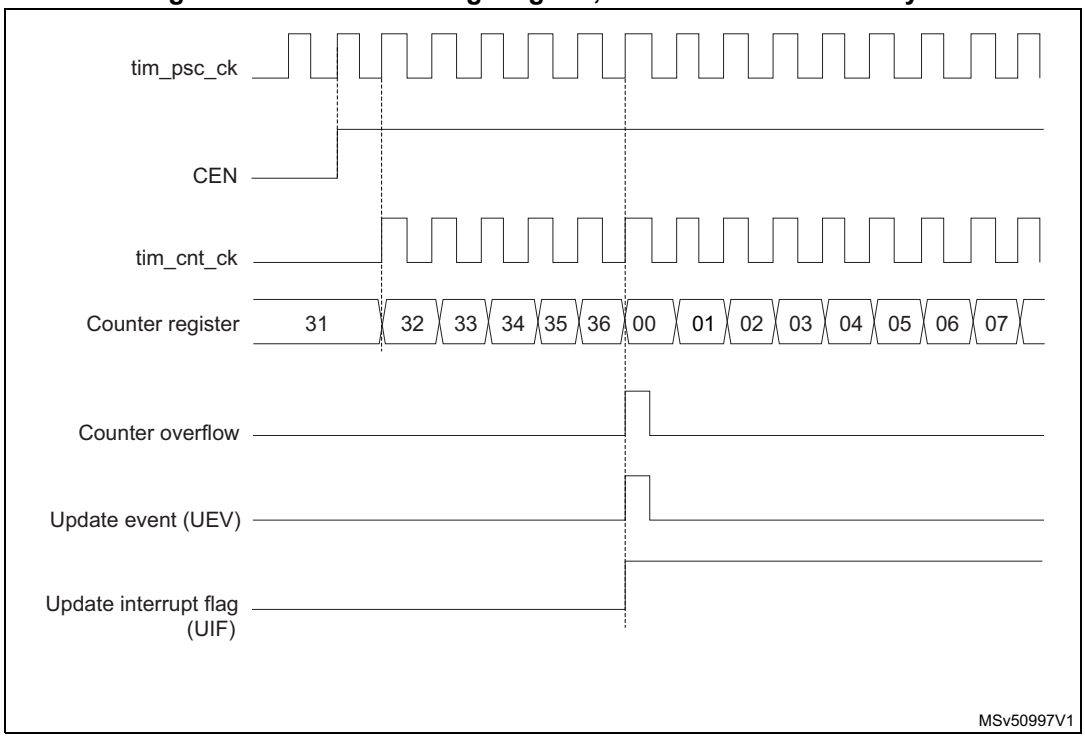


Figure 709. Counter timing diagram, internal clock divided by 2

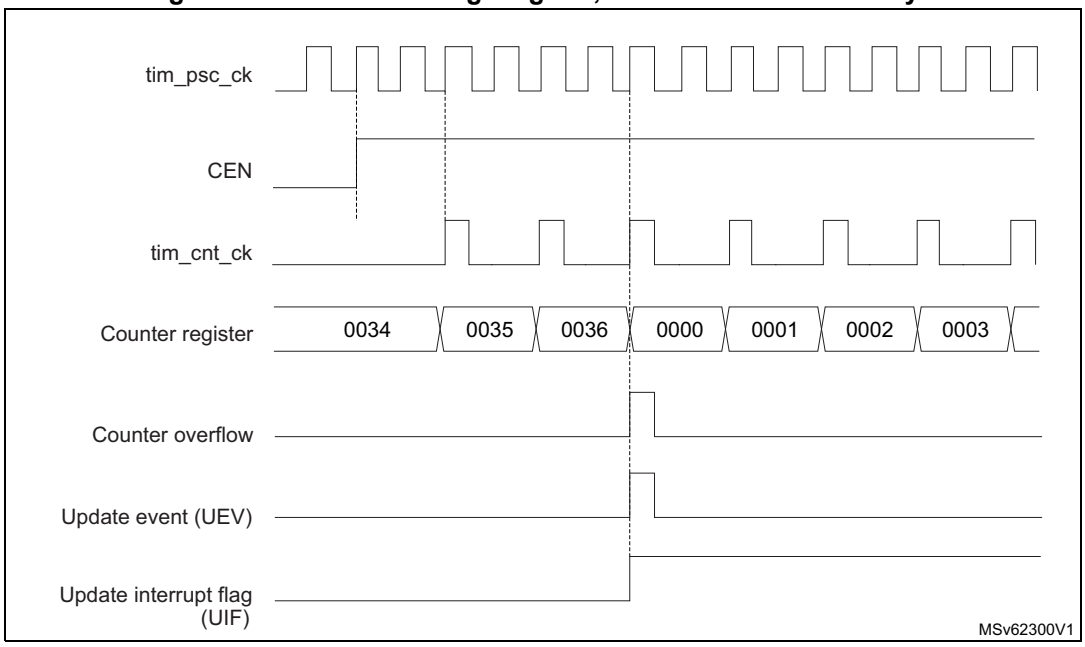


Figure 710. Counter timing diagram, internal clock divided by 4

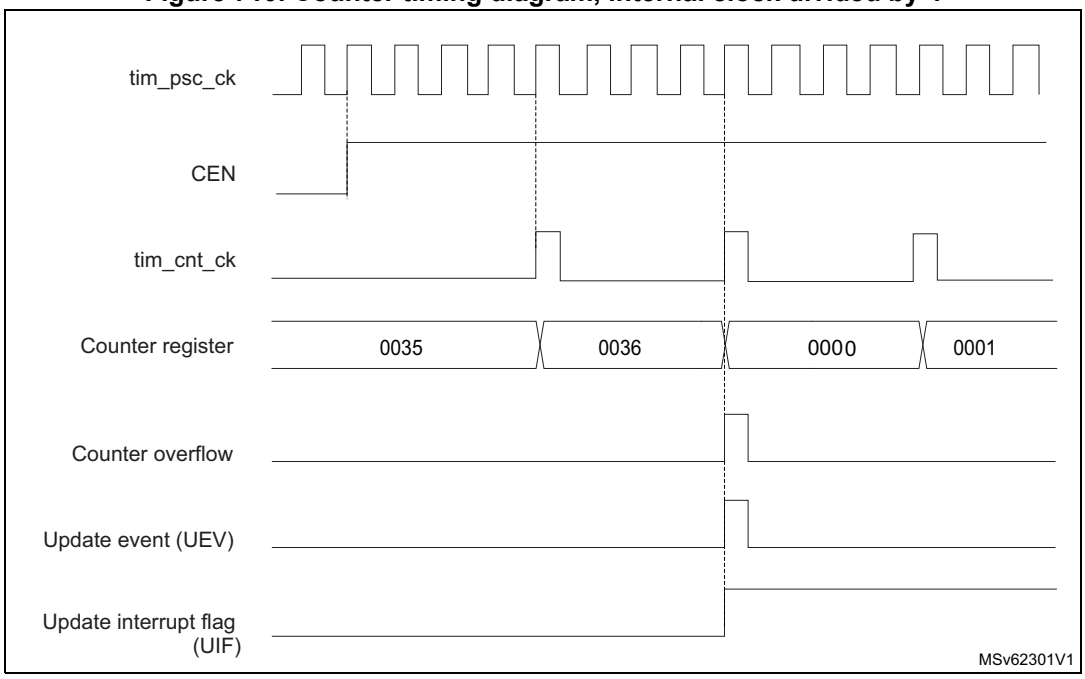


Figure 711. Counter timing diagram, internal clock divided by N

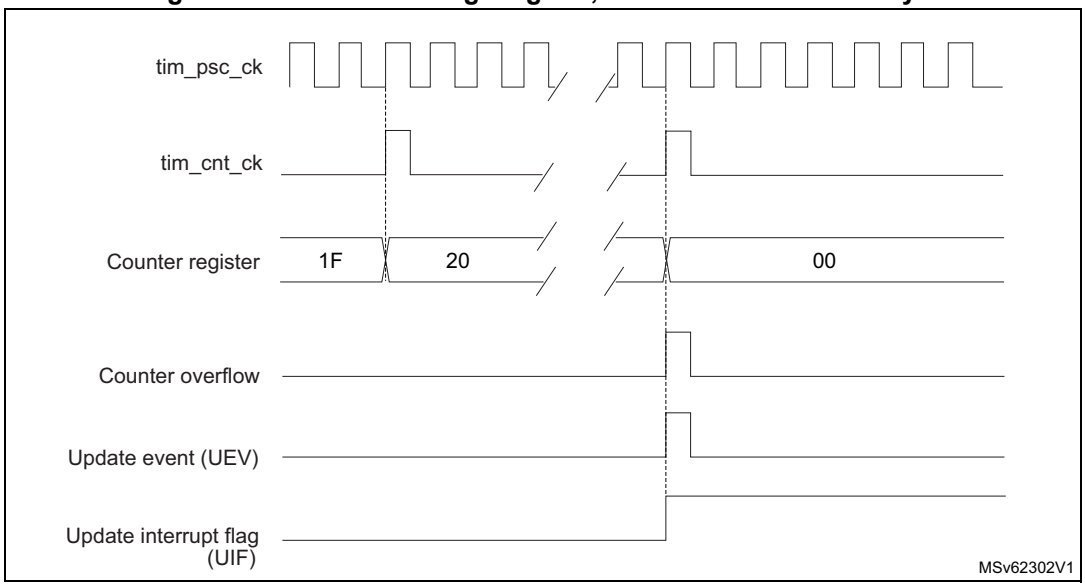


Figure 712. Counter timing diagram, update event when ARPE = 0 (TIM_ARR not preloaded)

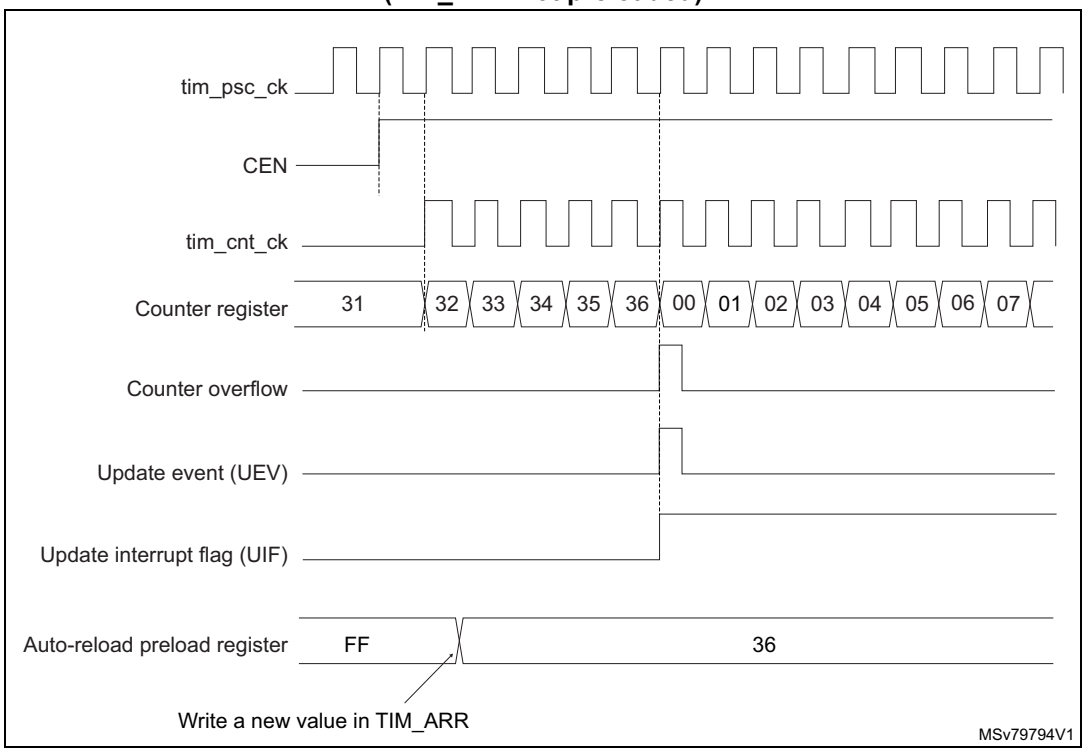
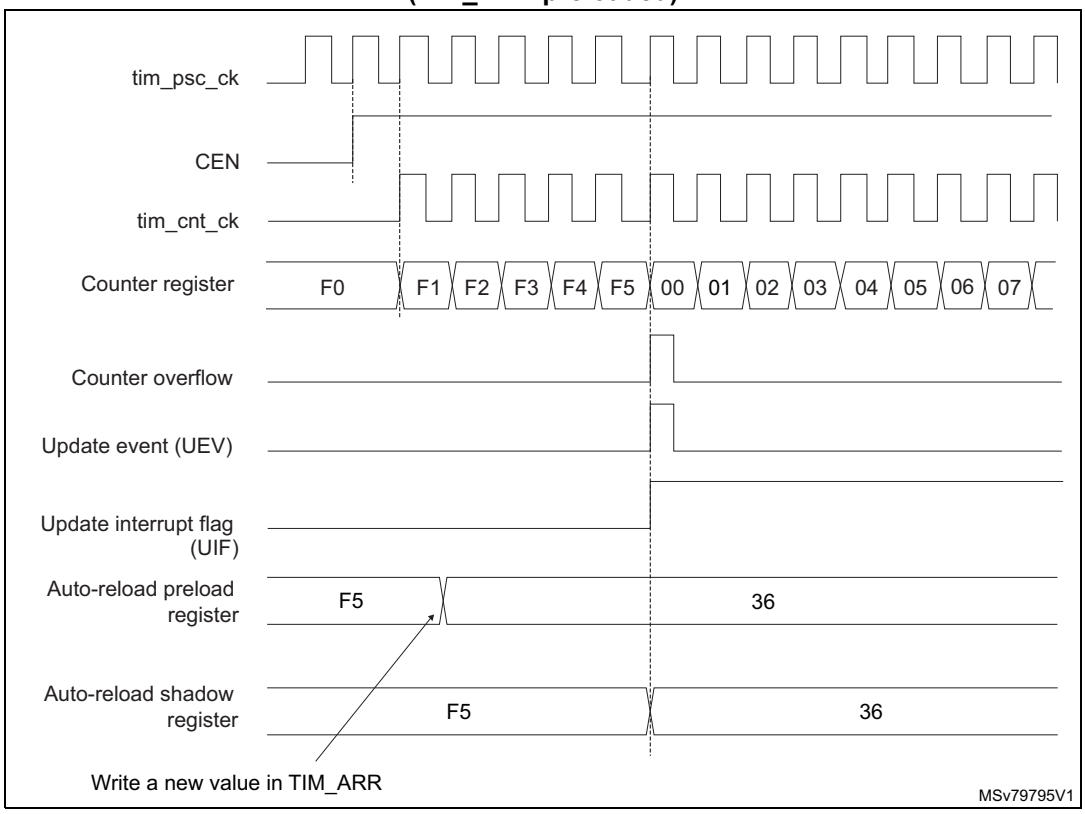


Figure 713. Counter timing diagram, update event when ARPE = 1 (TIM_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIM_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIM_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register).
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIM_ARR = 0x36.

Figure 714. Counter timing diagram, internal clock divided by 1

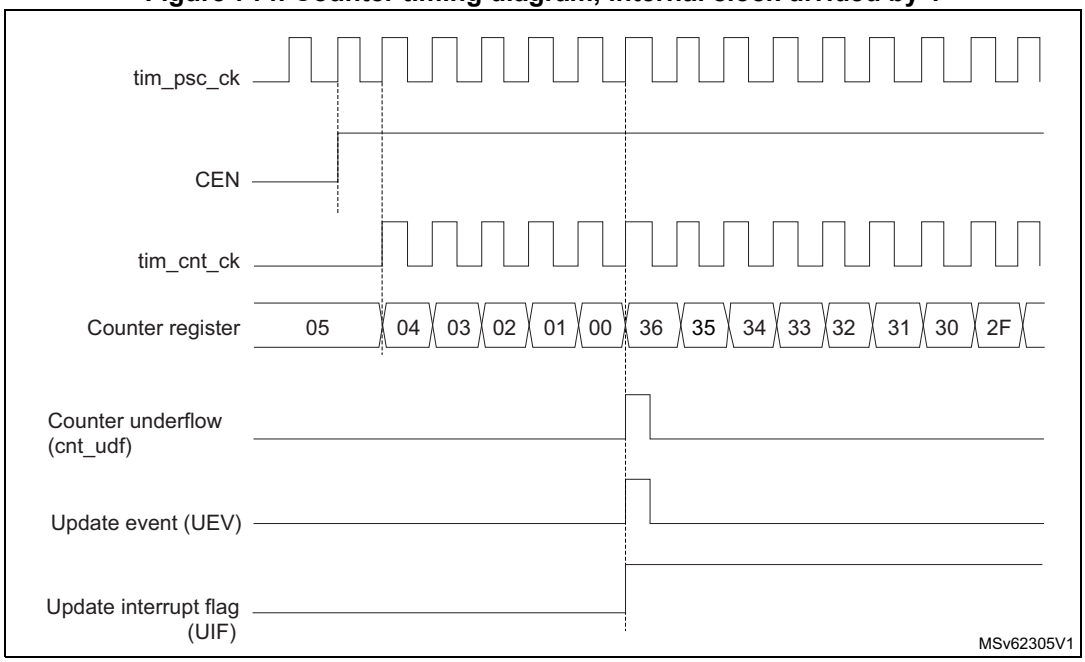


Figure 715. Counter timing diagram, internal clock divided by 2

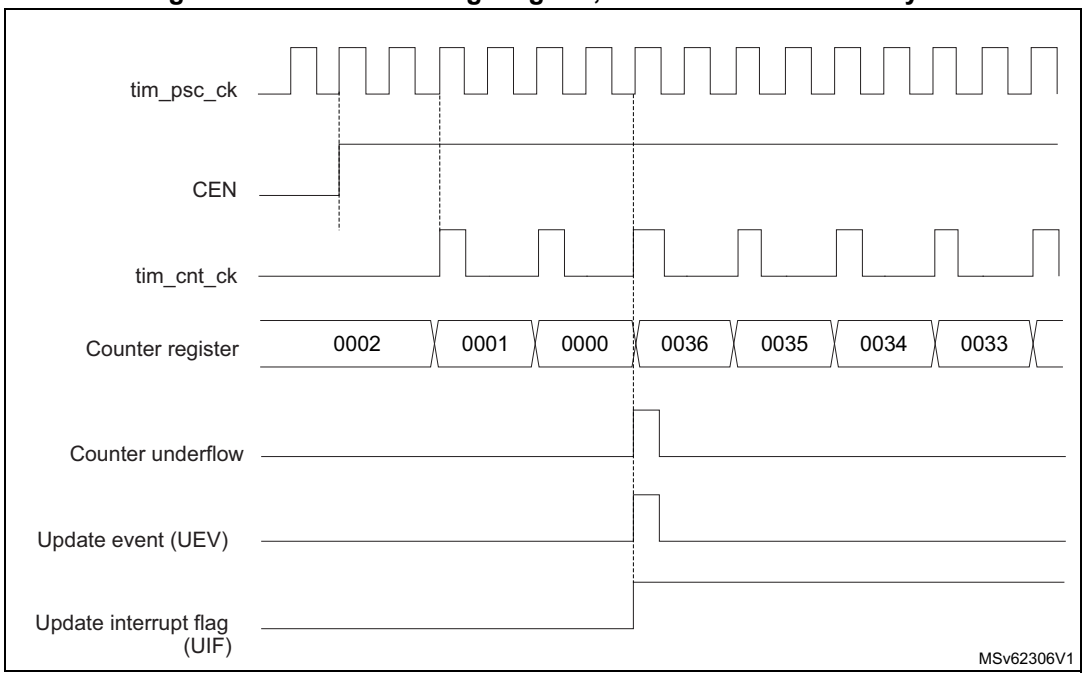


Figure 716. Counter timing diagram, internal clock divided by 4

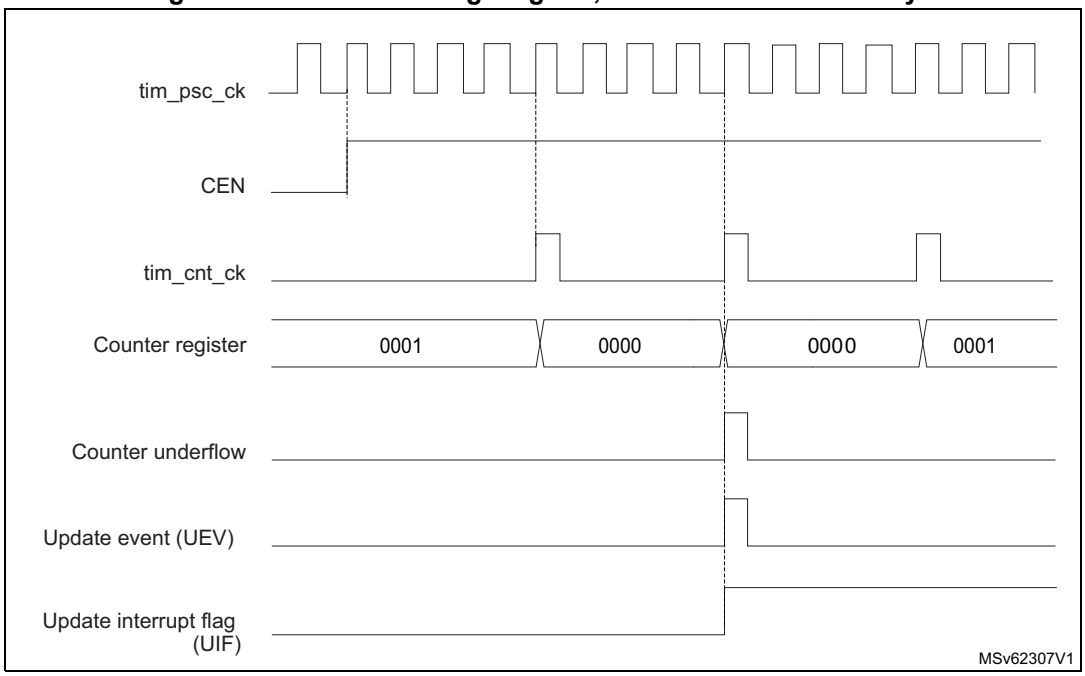


Figure 717. Counter timing diagram, internal clock divided by N

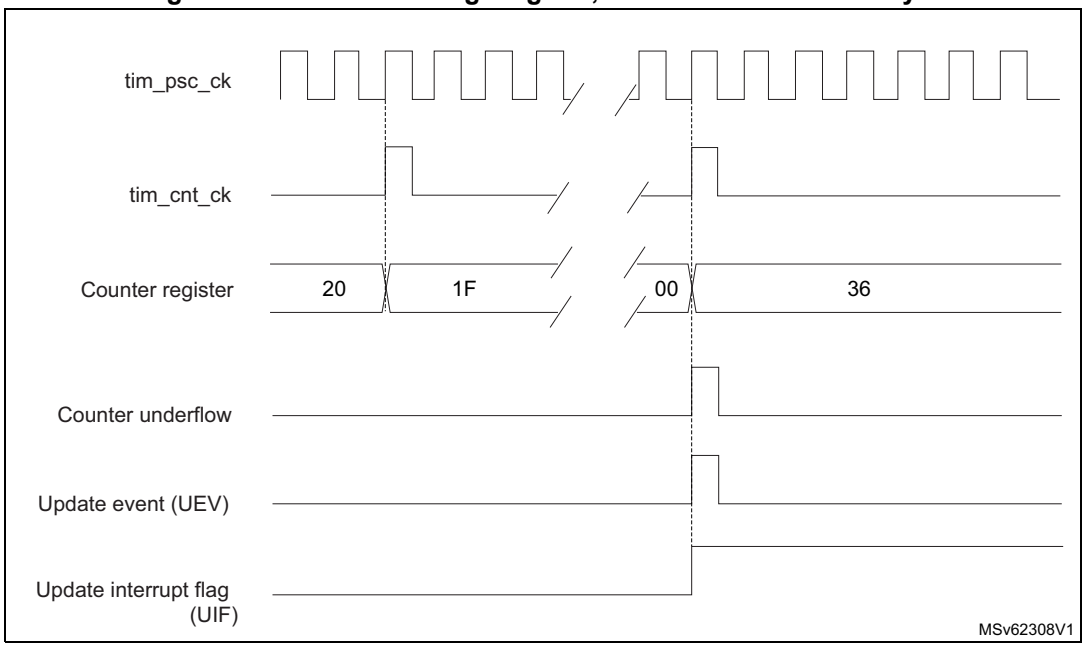
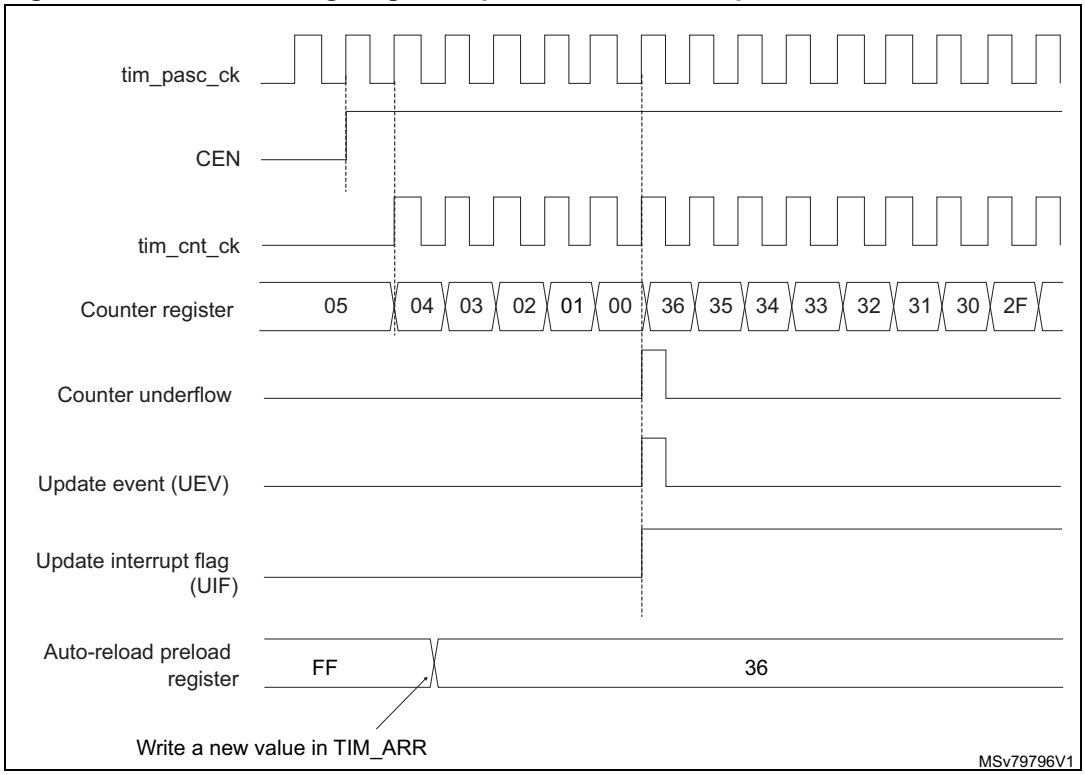


Figure 718. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIM_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIM_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIM_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIM_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIM_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

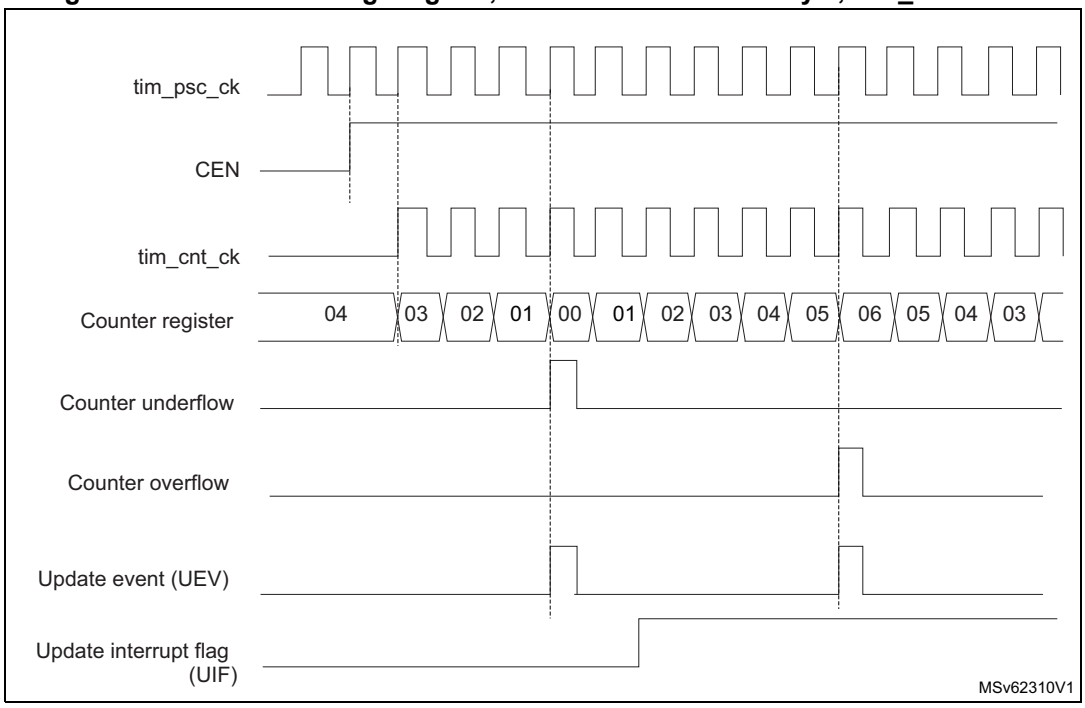
In addition, if the URS bit (update request selection) in TIM_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIM_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIM_PSC register)
- The autoreload active register is updated with the preload value (content of the TIM_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 719. Counter timing diagram, internal clock divided by 1, TIM_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 41.6: TIM1/TIM8/TIM20 registers*).

Figure 720. Counter timing diagram, internal clock divided by 2

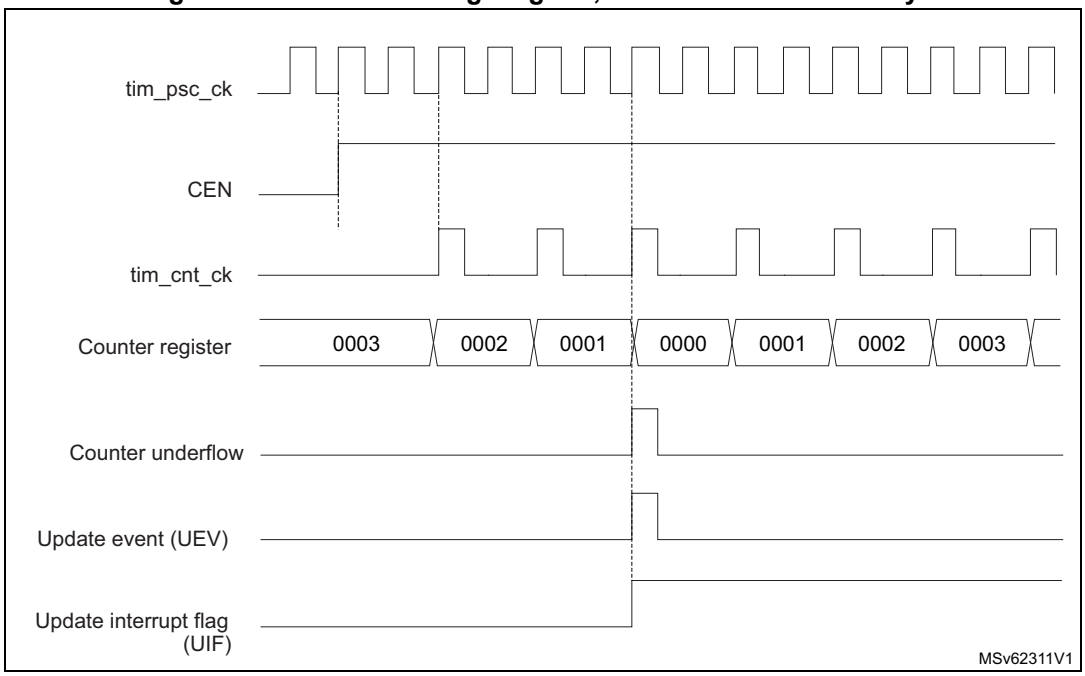


Figure 721. Counter timing diagram, internal clock divided by 4, TIM_ARR = 0x36

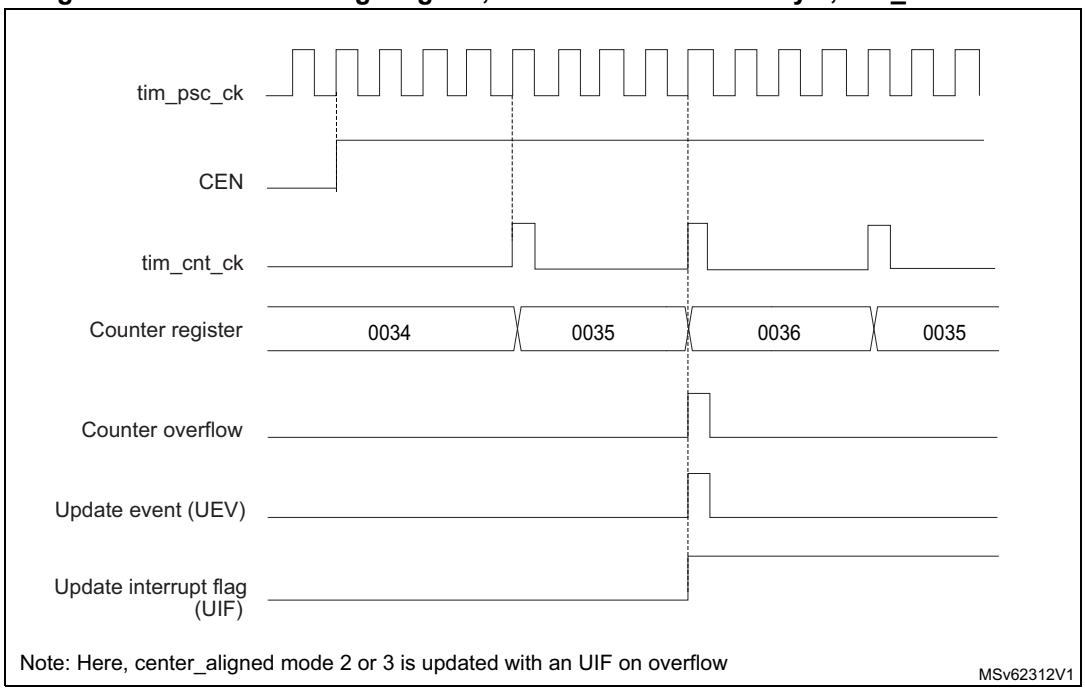


Figure 722. Counter timing diagram, internal clock divided by N

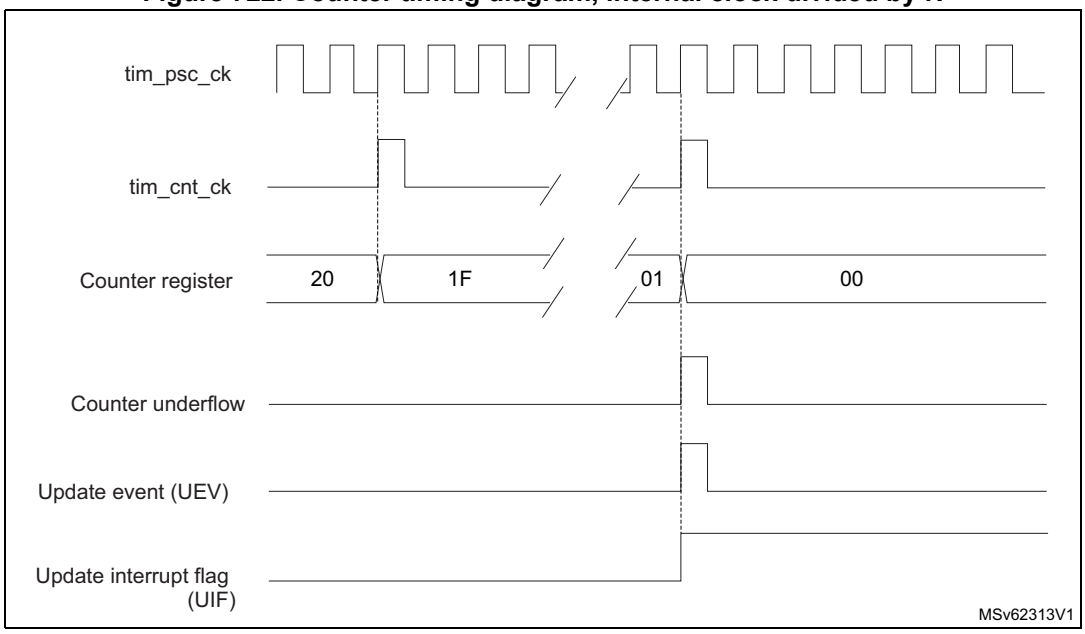


Figure 723. Counter timing diagram, update event with ARPE = 1 (counter underflow)

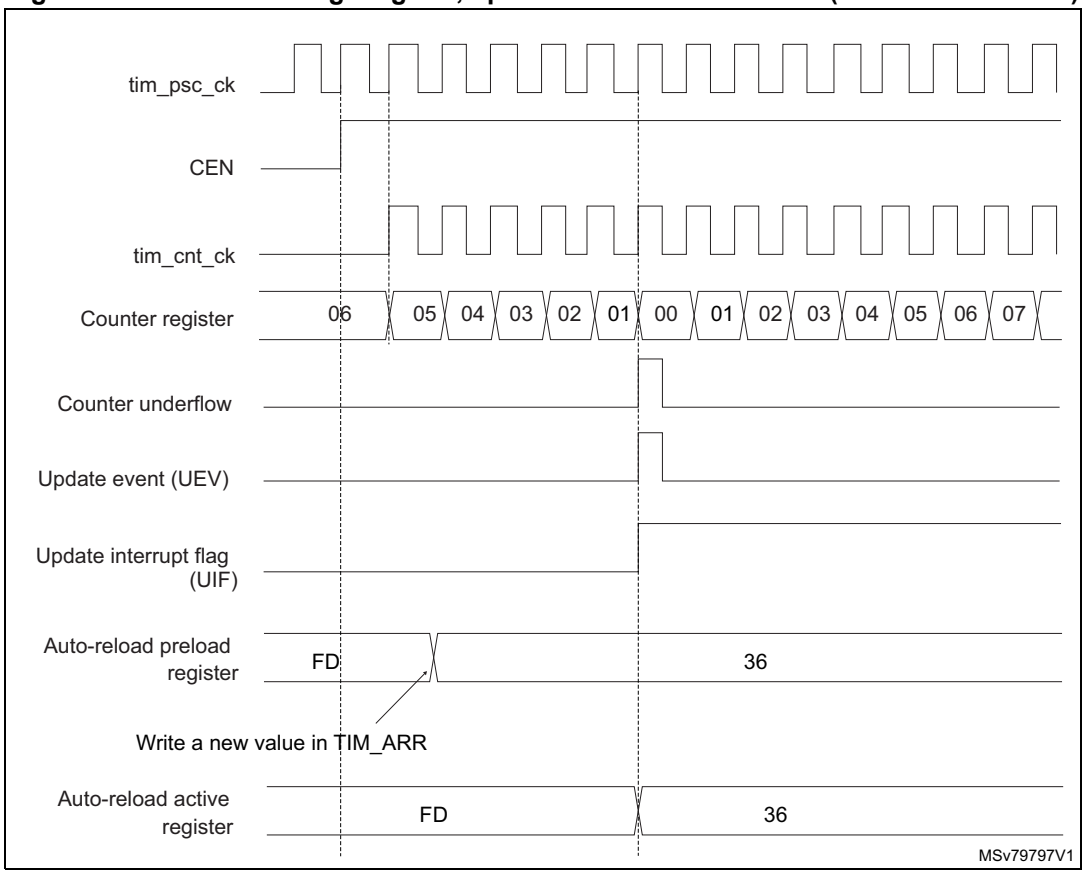
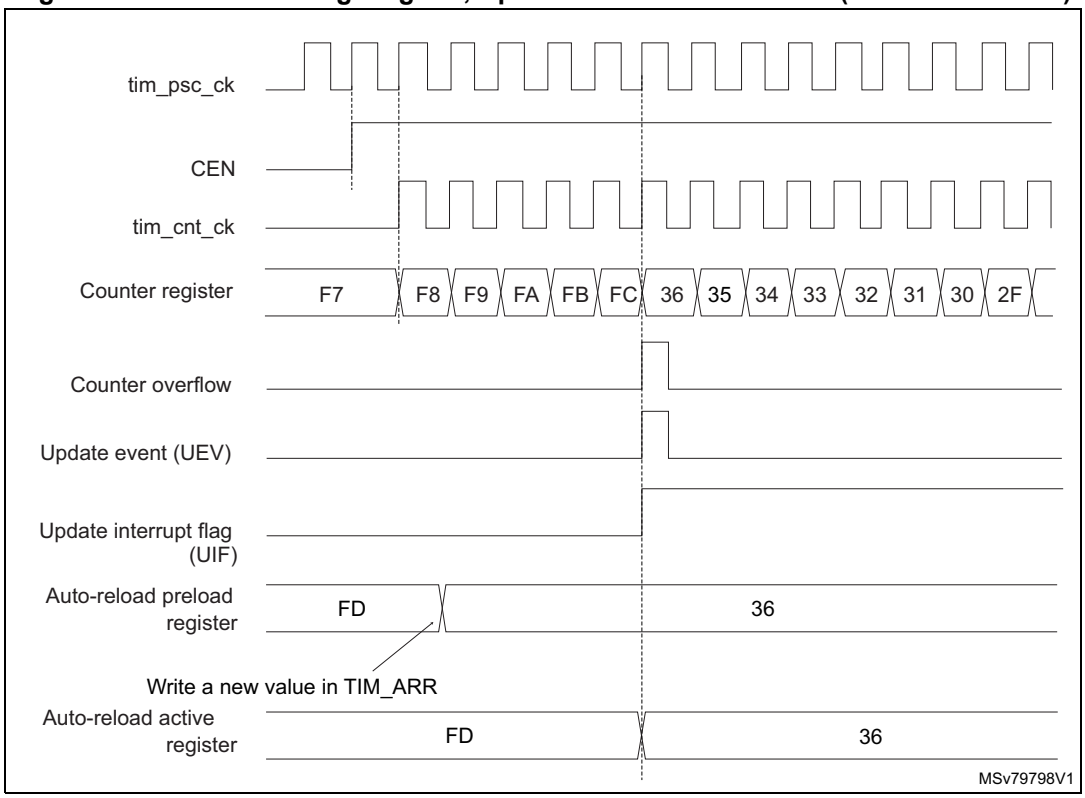


Figure 724. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



45.4.5 Repetition counter

Section 45.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIM_ARR autoreload register, TIM_PSC prescaler register, but also TIM_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIM_RCR repetition counter register.

The repetition counter is decremented:

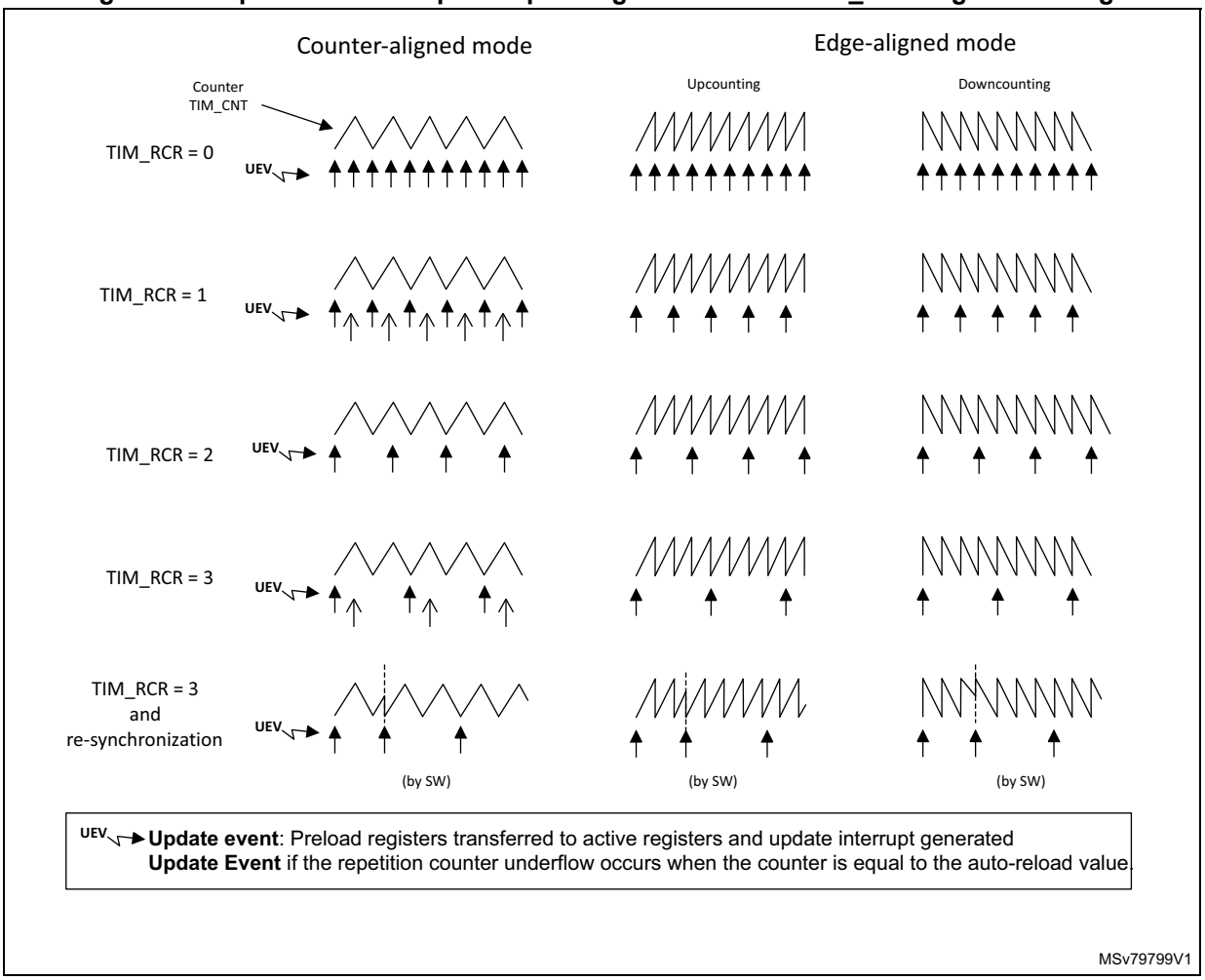
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIM_RCR register value (refer to *Figure 725*). When the update event is generated by software (by setting the UG bit in TIM_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIM_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 725. Update rate examples depending on mode and TIM_RCR register settings



45.4.6 Clock selection

The counter clock can be provided by the following clock sources:

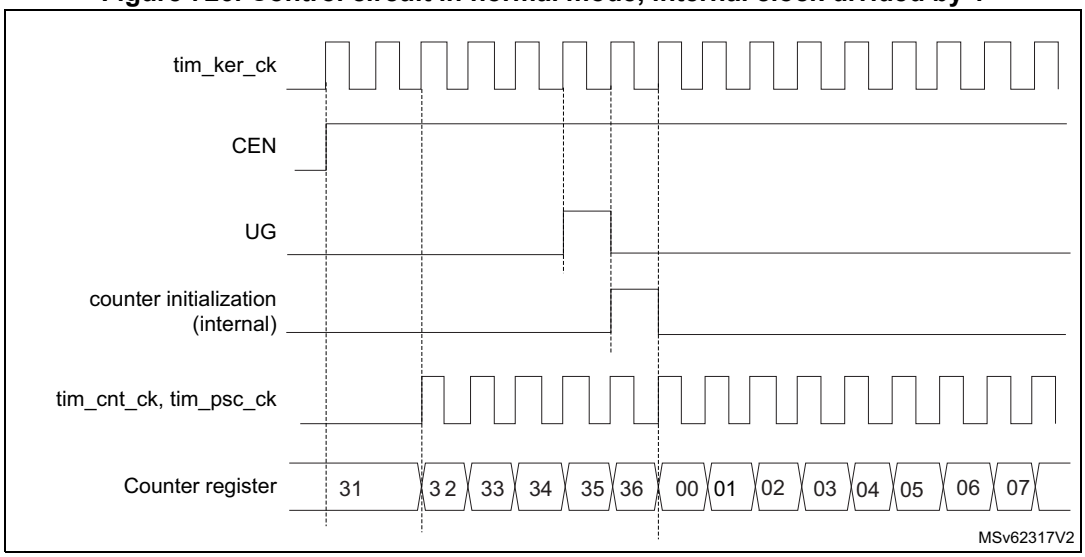
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer to enable another timer* for more details.
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 00000), then the CEN (in the TIM_CR1 register) and UG bits (in the TIM_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 726 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

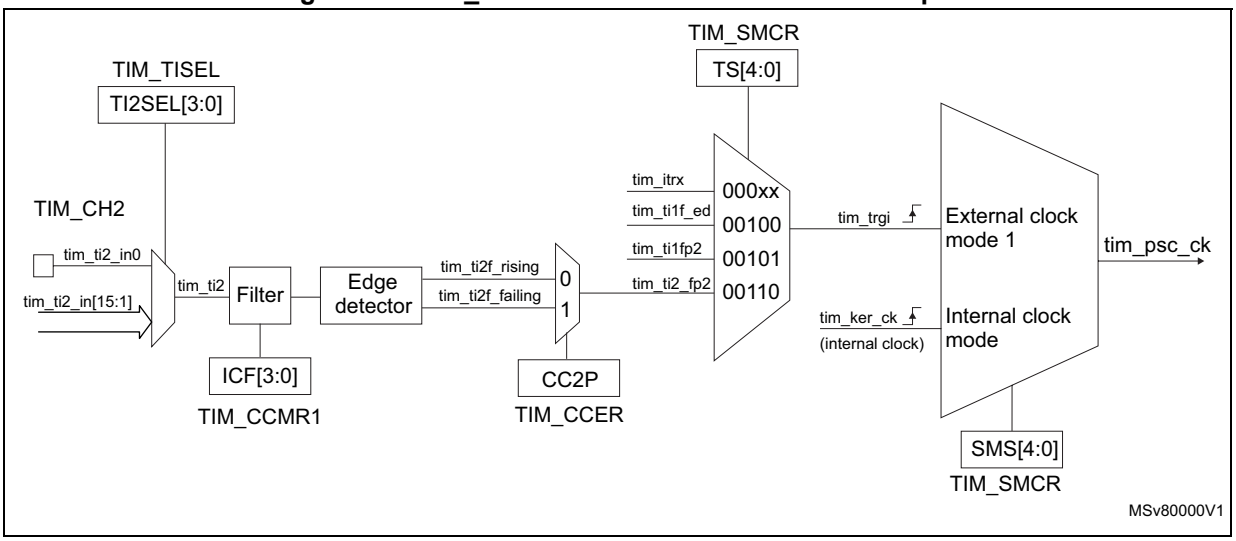
Figure 726. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 00111 in the TIM_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 727. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

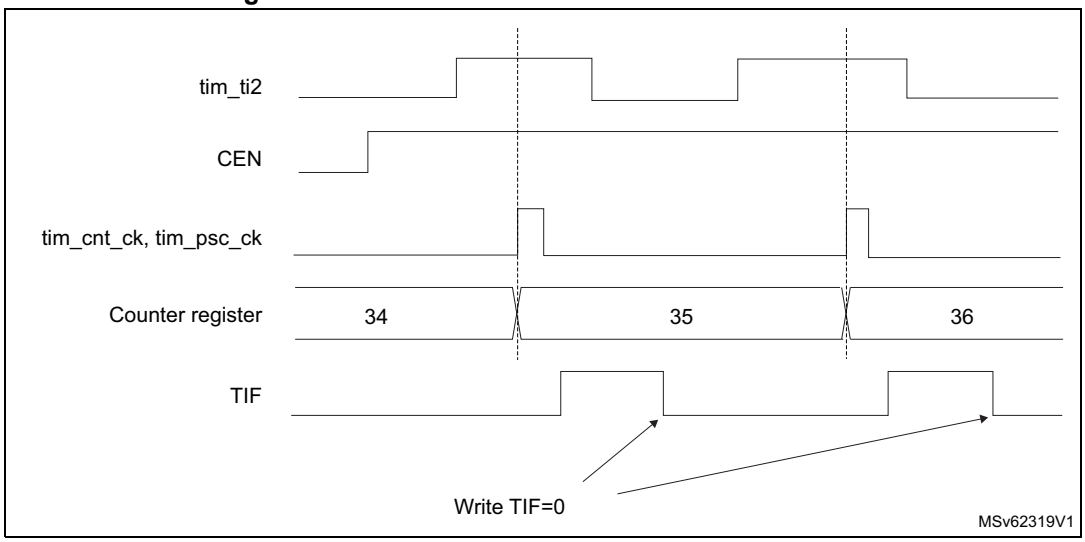
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIM_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIM_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 00111 in the TIM_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIM_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIM_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 728. Control circuit in external clock mode 1



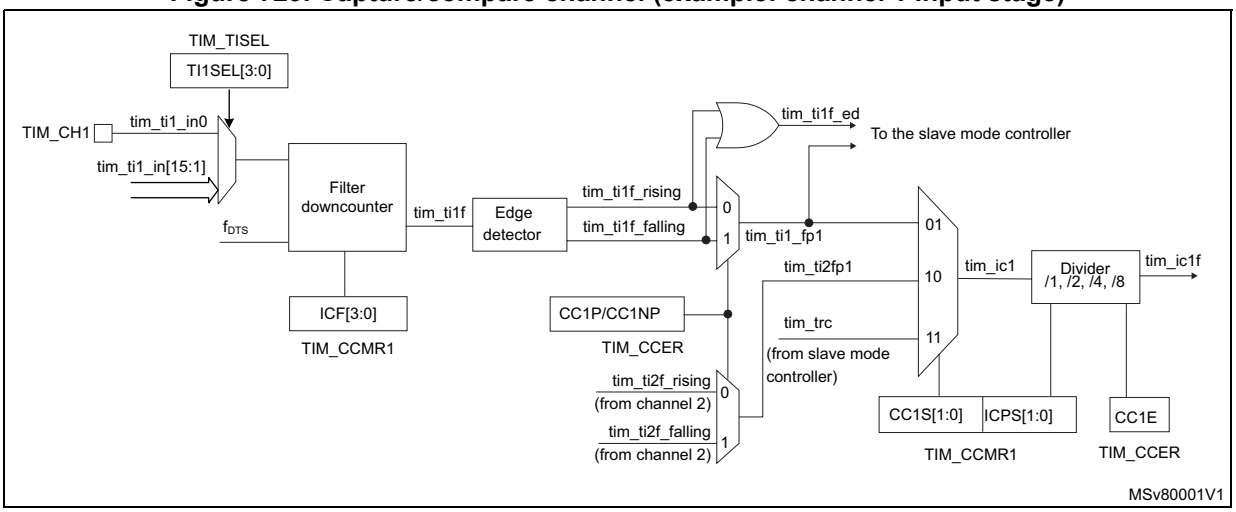
45.4.7 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 729 to *Figure 732* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 729. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 730. Capture/compare channel 1 main circuit

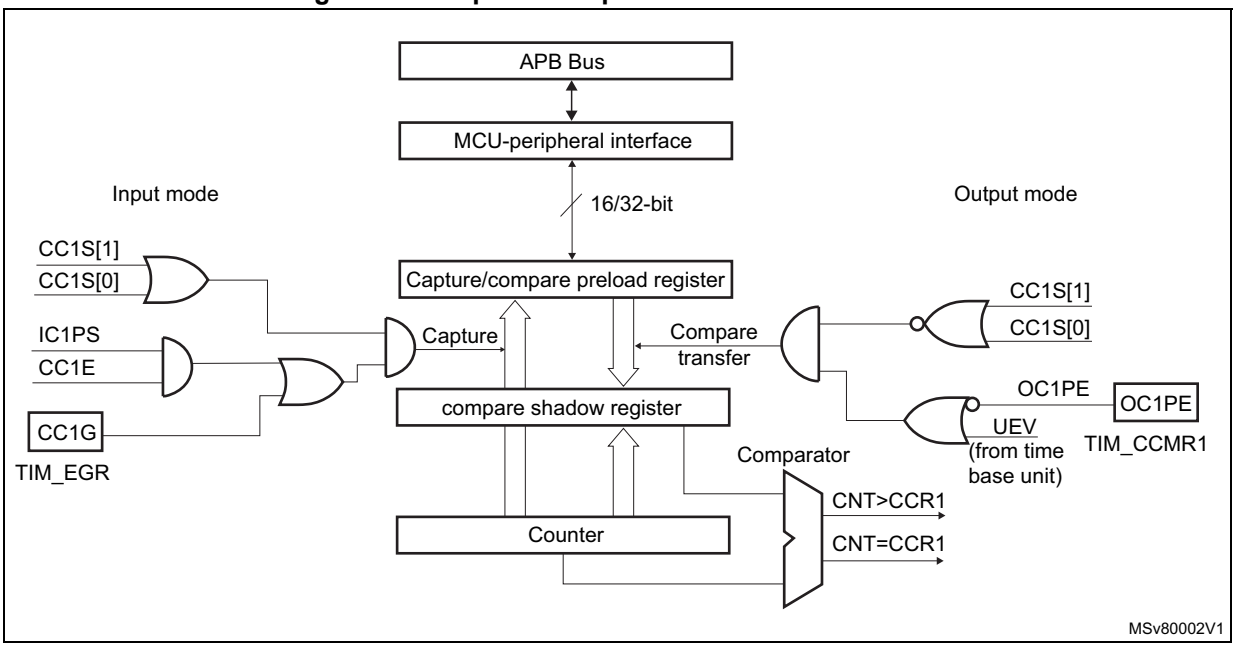


Figure 731. Output stage of capture/compare channel (channel 1)

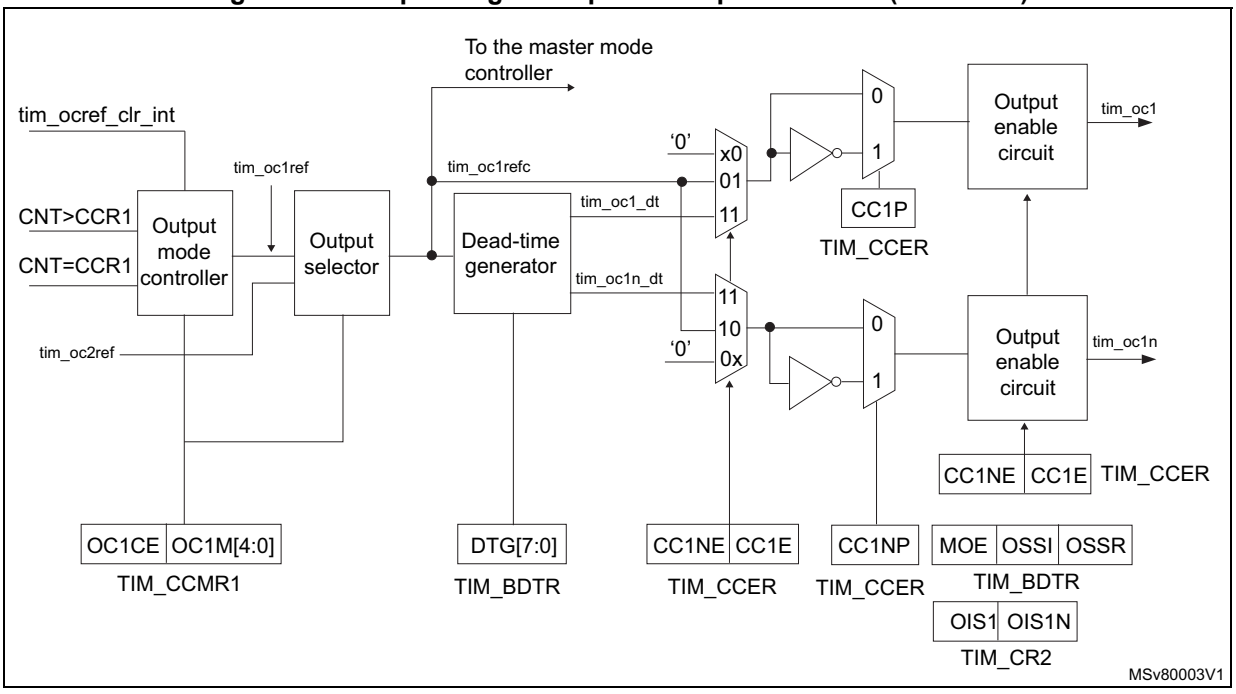
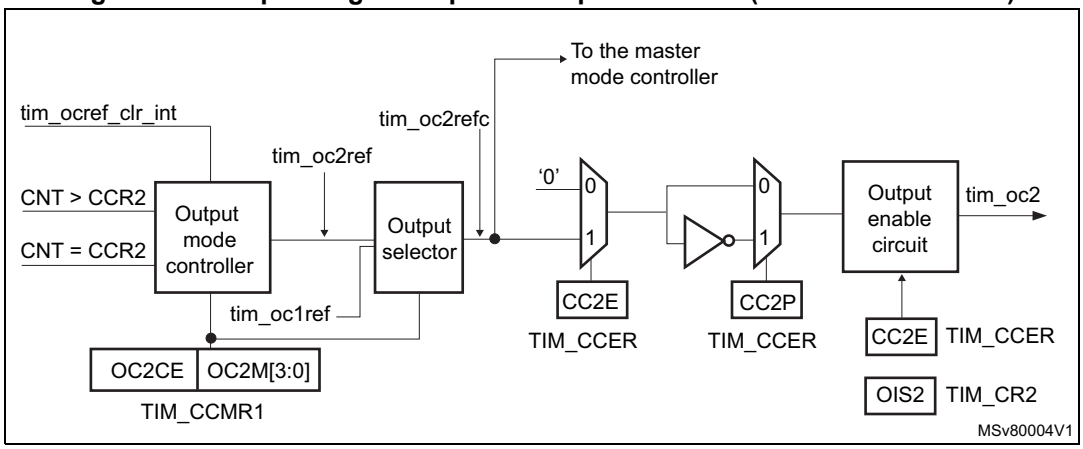


Figure 732. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

45.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIM_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIM_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIM_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input: TIM_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIM_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIM_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIM_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new

level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIM_CCMR1 register.

4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIM_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIM_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM_DIER register, and/or the DMA request by setting the CC1DE bit in the TIM_DIER register.

When an input capture occurs:

- The TIM_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIM_EGR register.

45.4.9 PWM input mode (only for TIM15)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIM_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

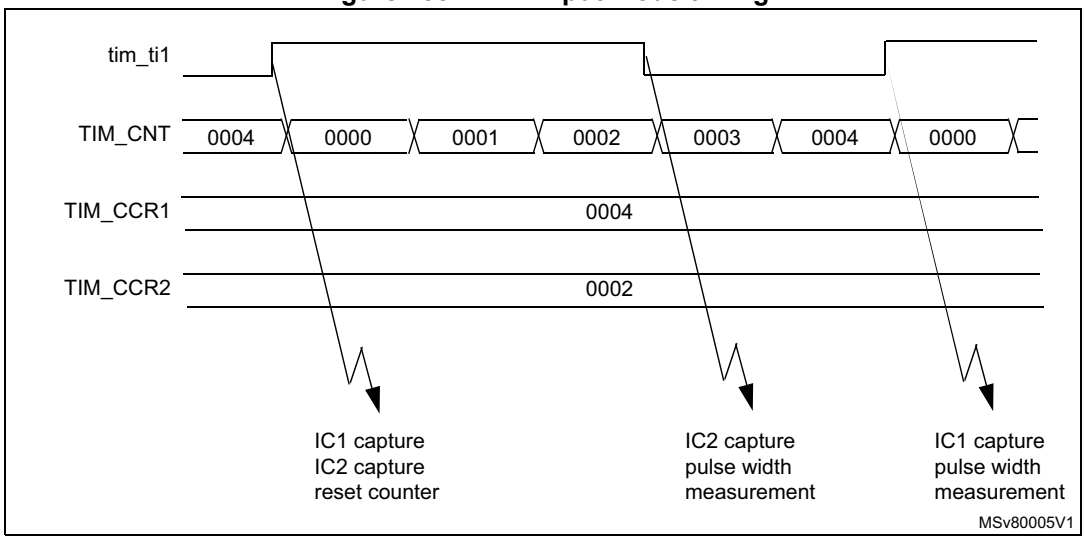
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIM_CCR1 register) and the duty cycle (in TIM_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIM_TISEL register.
2. Select the active input for TIM_CCR1: write the CC1S bits to 01 in the TIM_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIM_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
4. Select the active input for TIM_CCR2: write the CC2S bits to 10 in the TIM_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIM_CCR2): write the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIM_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 00100 in the TIM_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIM_CCER register.

Figure 733. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

45.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIM_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 00101 in the OCxM bits in the corresponding TIM_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 00100 in the TIM_CCMRx register.

Anyway, the comparison between the TIM_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

45.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIM_CCMRx register) and the output polarity (CCxP bit in the TIM_CCER register). The output pin can keep its level (OCXM = 00000), be set active (OCxM = 00001), be set inactive (OCxM = 00010) or can toggle (OCxM = 00011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIM_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIM_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIM_DIER register, CCDS bit in the TIM_CR2 register for the DMA request selection).

The TIM_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIM_CCMRx register.

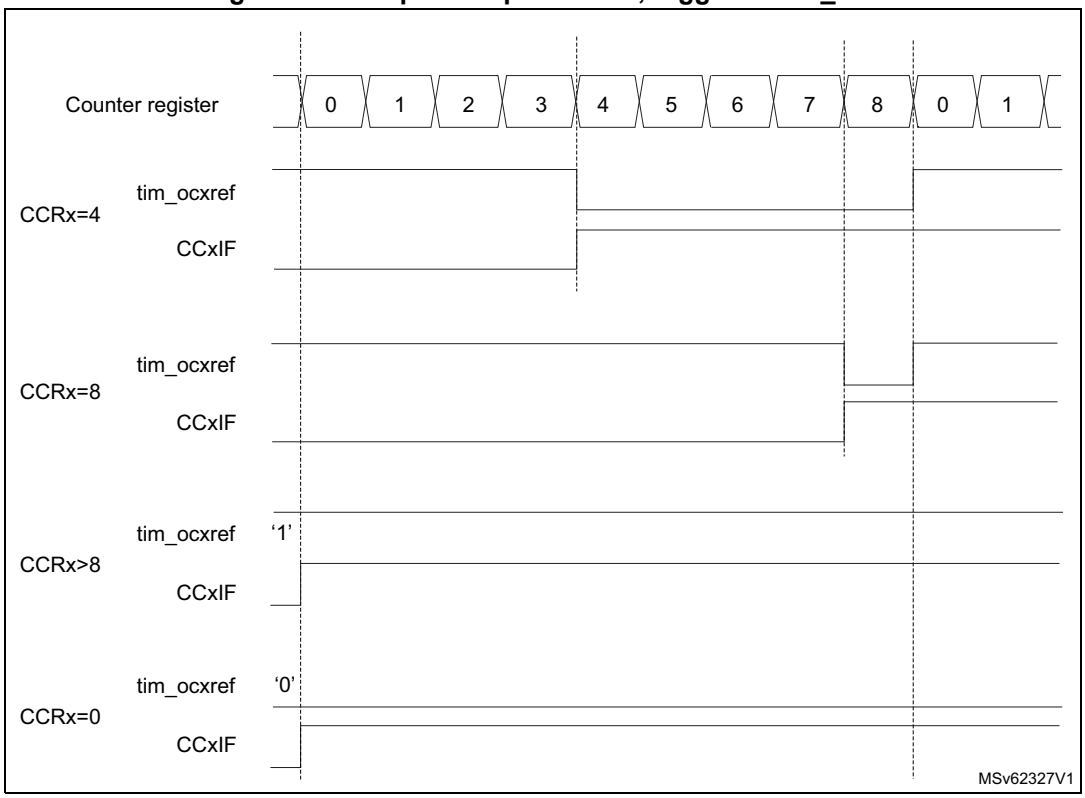
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIM_ARR and TIM_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 00011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIM_CR1 register.

The TIM_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIM_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 734*.

Figure 734. Output compare mode, toggle on tim_oc1



45.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIM_ARR register and a duty cycle determined by the value of the TIM_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 00110 (PWM mode 1) or 00111 (PWM mode 2) in the OCxM bits in the TIM_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIM_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIM_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIM_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIM_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIM_CCER and TIM_BDTR registers). Refer to the TIM_CCER register description for more details.

In PWM mode (1 or 2), TIM_CNT and TIM_CCRx are always compared to determine whether TIM_CCRx ≤ TIM_CNT or TIM_CNT ≤ TIM_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM_CR1 register.

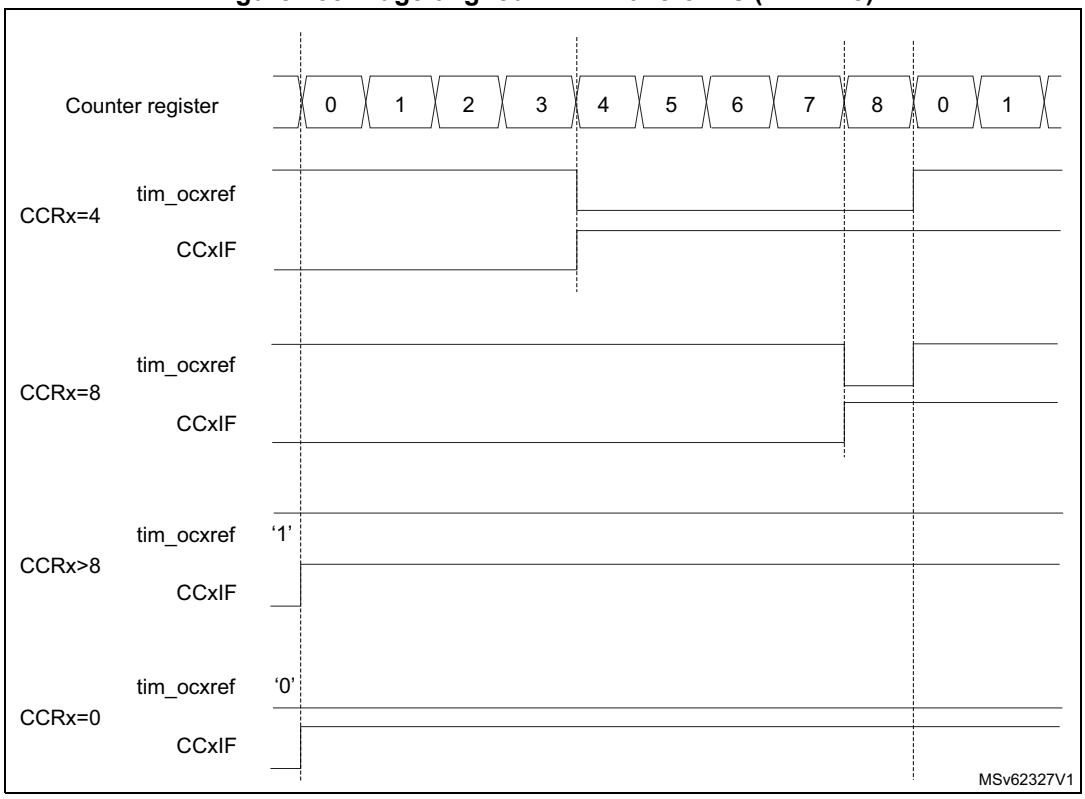
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIM_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIM_CNT < TIM_CCRx else it becomes low. If the compare value in TIM_CCRx is greater than the autoreload value (in TIM_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 735* shows some edge-aligned PWM waveforms in an example where TIM_ARR = 8.

Figure 735. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIM_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIM_CNT > TIM_CCRx else it becomes high. If the compare value in TIM_CCRx is greater than the autoreload value in TIM_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

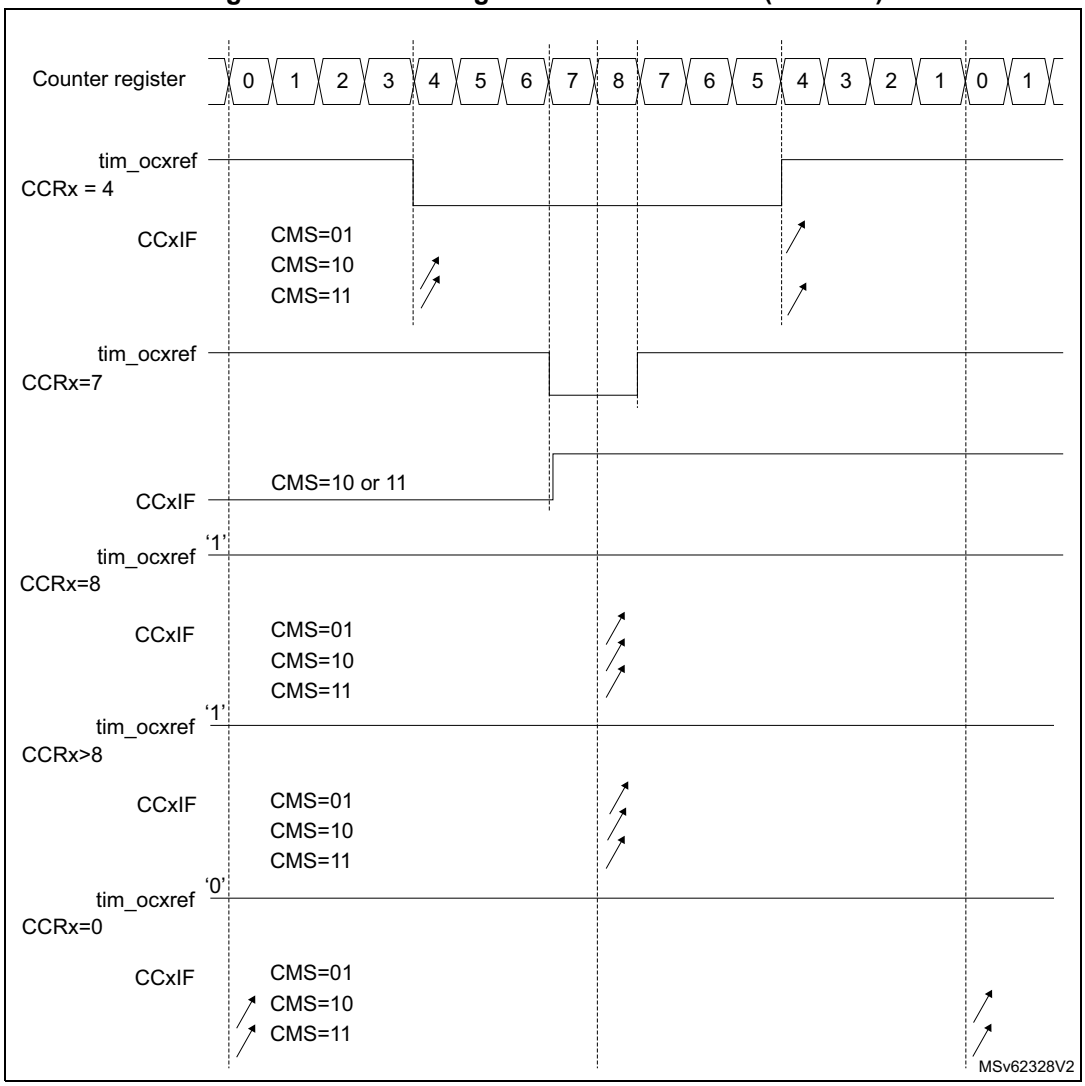
Center-aligned mode is active when the CMS bits in TIM_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIM_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 736 shows some center-aligned PWM waveforms in an example where:

- TIM_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIM_CR1 register.

Figure 736. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIM_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

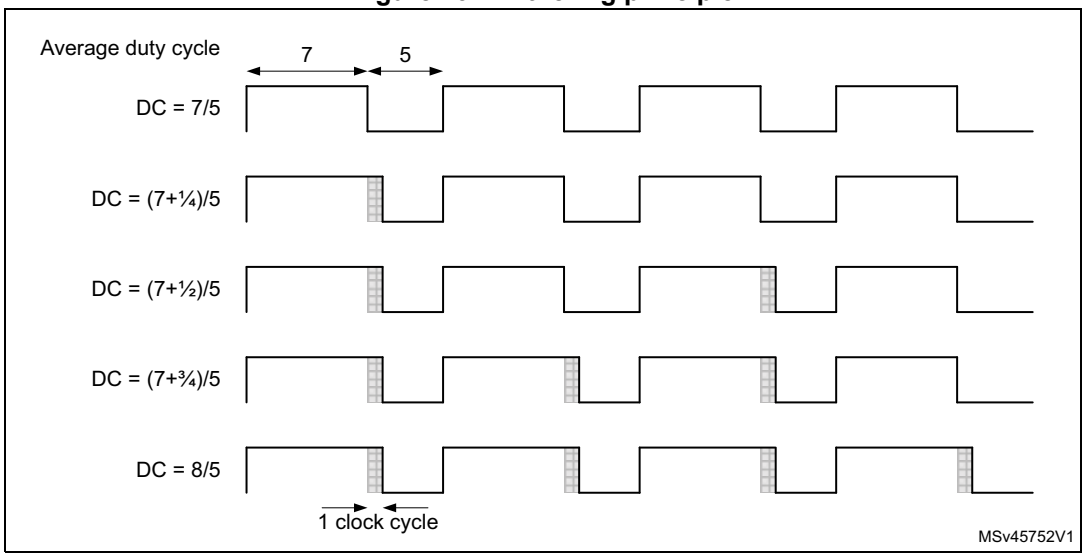
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIM_CNT > TIM_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIM_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIM_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 737* below presents the dithering principle applied to four consecutive PWM cycles.

Figure 737. Dithering principle

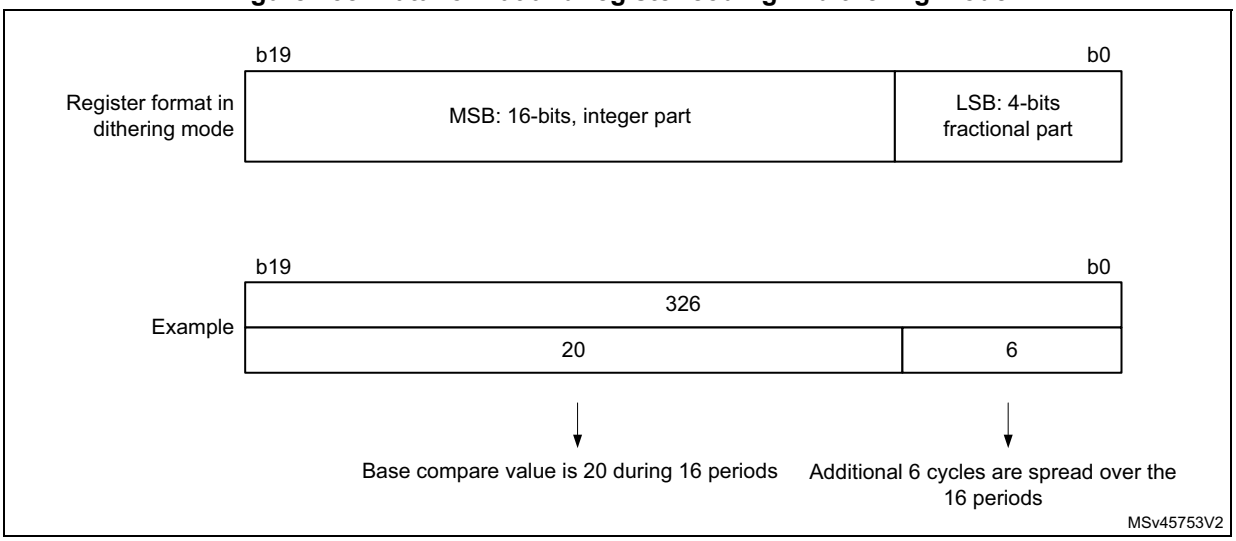


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 738* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 738. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

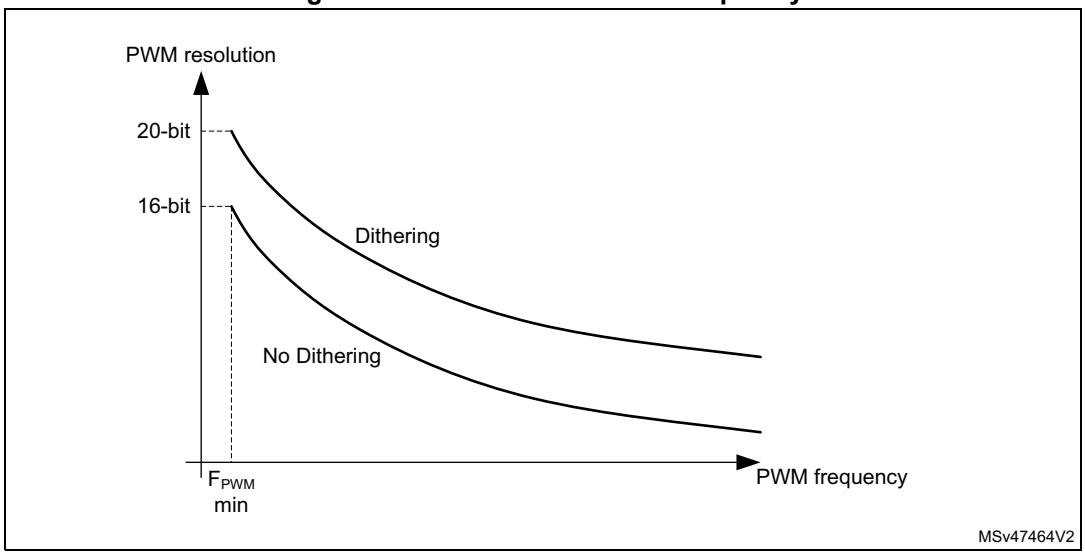
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIM_ARR and TIM_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

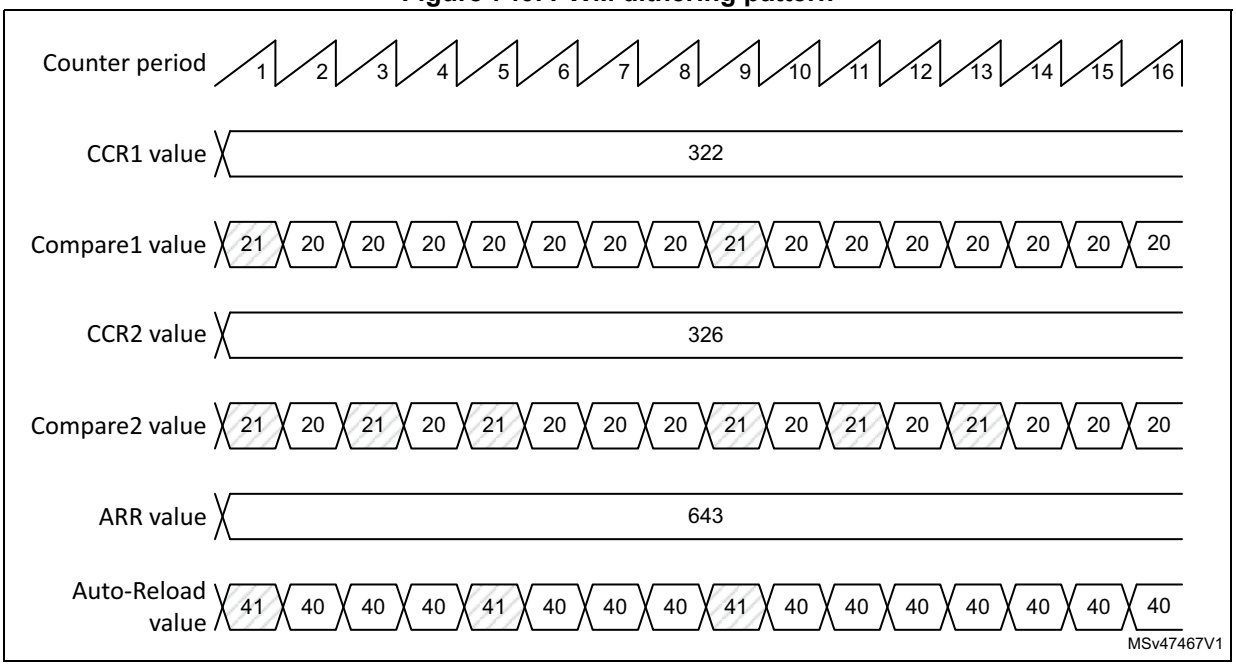
As shown on the *Figure 739* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 739. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 740* below.

Figure 740. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 522* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 522. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIM_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 741*.

Figure 741. Dithering effect on duty cycle in center-aligned PWM mode

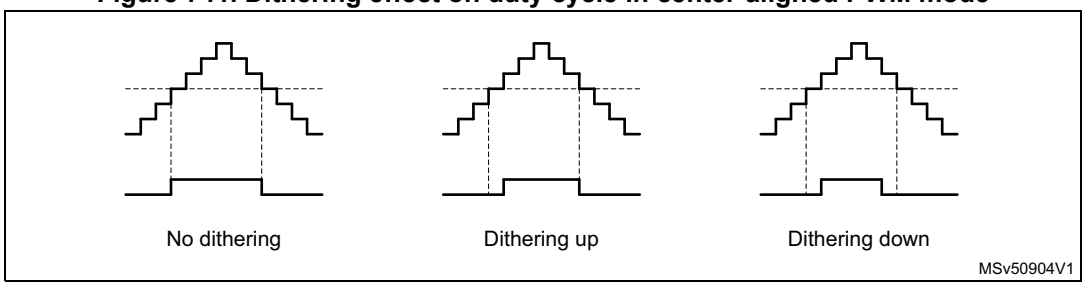


Table 523 shows how the dithering pattern is added in center-aligned PWM mode.

Table 523. CCR register change dithering pattern in center-aligned PWM mode

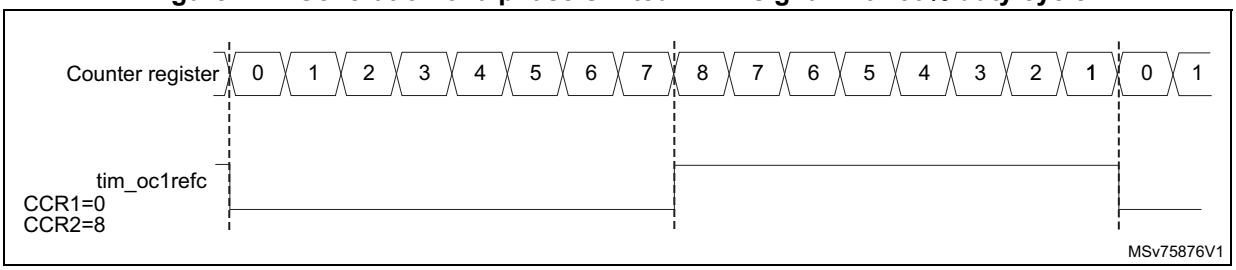
LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

45.4.13 Asymmetric PWM mode (TIM15 only)

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and the phase-shift are determined by a pair of TIM_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle. For instance, tim_oc1refc is controlled by TIM_CCR1 and TIM_CCR2.

Figure 742 represents an example of signal that can be generated using asymmetric PWM mode (channels 1 and 2 are configured in asymmetric PWM mode 2).

Figure 742. Generation of a phase-shifted PWM signal with 50% duty cycle



Multiple asymmetric PWM modes are available to combine different pairs of TIM_CCRx registers and generate the required waveform, as listed in *Table 524*. The selection is done with the OCxM[4:0] bitfield in TIM_CCMRx registers. For each pair of TIM_CCRx registers, two options are given (for example asymmetric PWM mode 3 and 4), one being in PWM mode 1 (channel active as long as the counter is below the TIM_CCRx value), the other one in PWM mode 2 (channel active when the counter is above the TIM_CCRx value).

For the asymmetric PWM modes 1 and 2, it is mandatory to have the same PWM mode for both channels. For instance, it is mandatory to have OC1M[4:0] = OC2M[4:0] = 01110 for the asymmetric PWM mode 1 on oc1refc.

The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bits are not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if a tim_oc1refc signal is generated on channel 1 (asymmetric PWM mode 1), it is possible to output either the regular tim_oc2ref signal on channel 2 or a tim_oc2refc signal resulting from asymmetric PWM mode 1.

Table 524. Asymmetric PWM modes

Output compare mode	OCxM[4:0]	oc1refc	oc2refc	PWM Mode(1)
Asymmetric PWM mode 1	01110	oc1ref (up-counting)		1
Asymmetric PWM mode 2	01111	oc2ref (down-counting)		2

1. PWM mode 1: channel active as long as the counter is below the TIM_CCRx value.
PWM mode 2: channel active when the counter is above the TIM_CCRx value. Unless explicitly mentioned, the PWM mode must be identical for the 2 channels used for the asymmetric mode.

45.4.14 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIM_ARR register, the duty cycle and delay are determined by the two TIM_CCRx registers. Both signals, tim_oc1refc and tim_oc2refc, are made of an OR or AND logical combination of two reference PWMs controlled respectively by TIM_CCR1 and TIM_CCR2.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 01100 (Combined PWM mode 1) or 01101 (Combined PWM mode 2) in the OCxM bits in the TIM_CCMRx register.

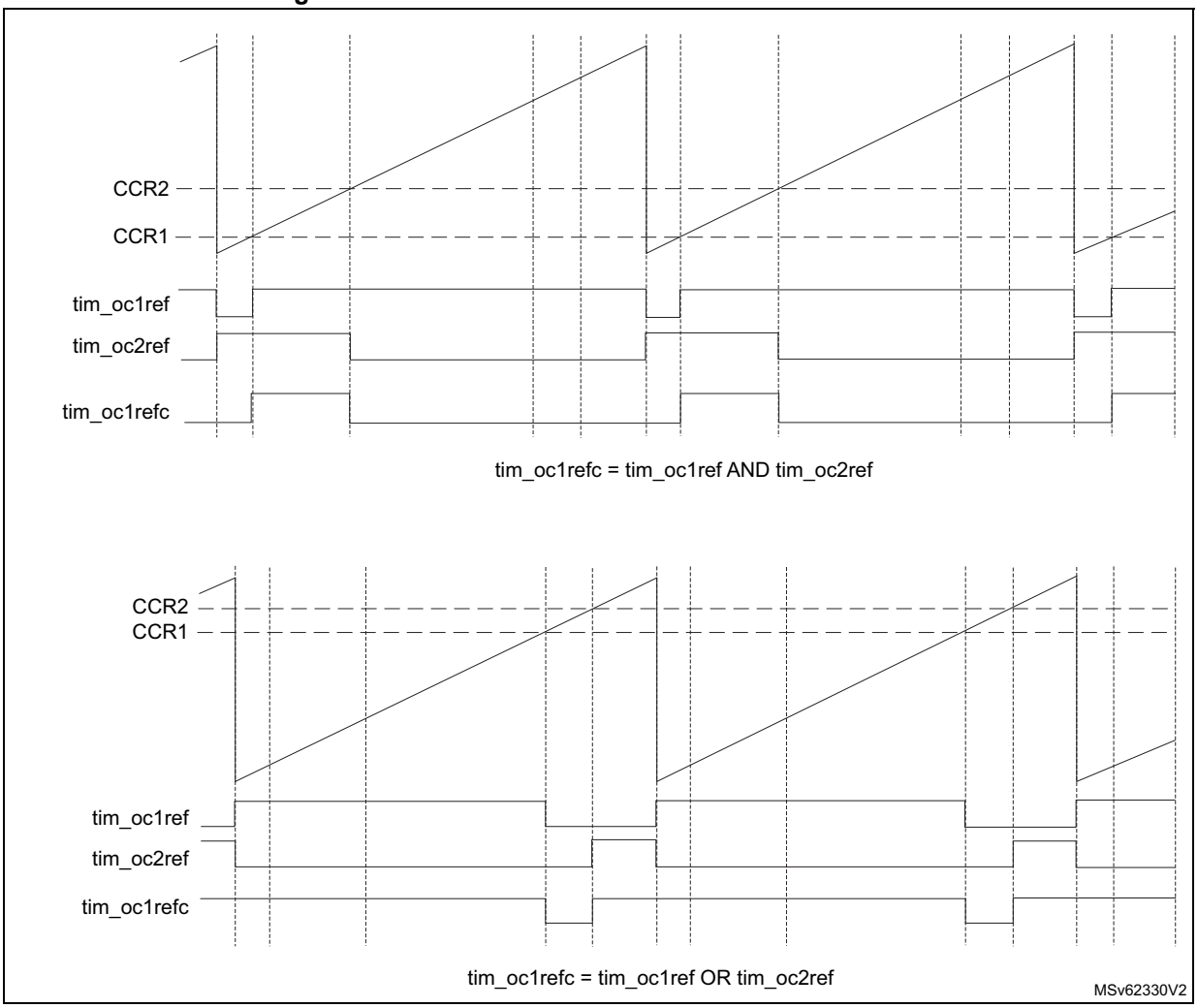
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[4:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 743 represents an example of signals that can be generated using Combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 743. Combined PWM mode on channel 1 and 2



45.4.15 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIM_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIM_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIM_BDTR and TIM_CR2 registers. Refer to *Table 531: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) on page 2384* and *Table 533: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 2416* for more details. In

particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

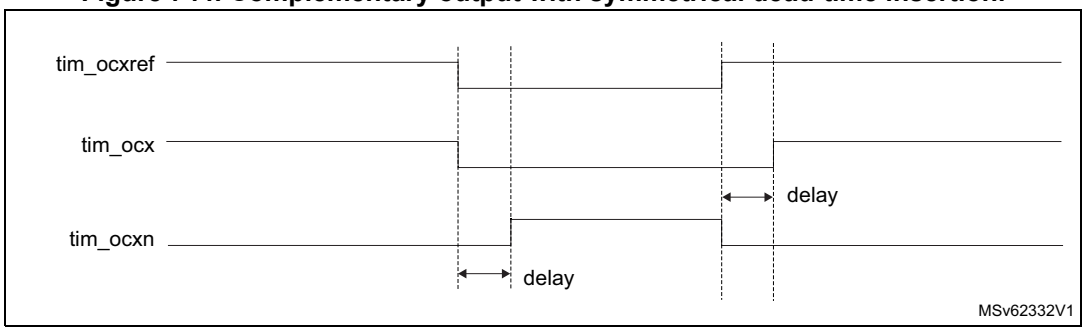
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 744. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIM_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 745*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIM_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIM_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIM_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 745. Asymmetrical deadtime

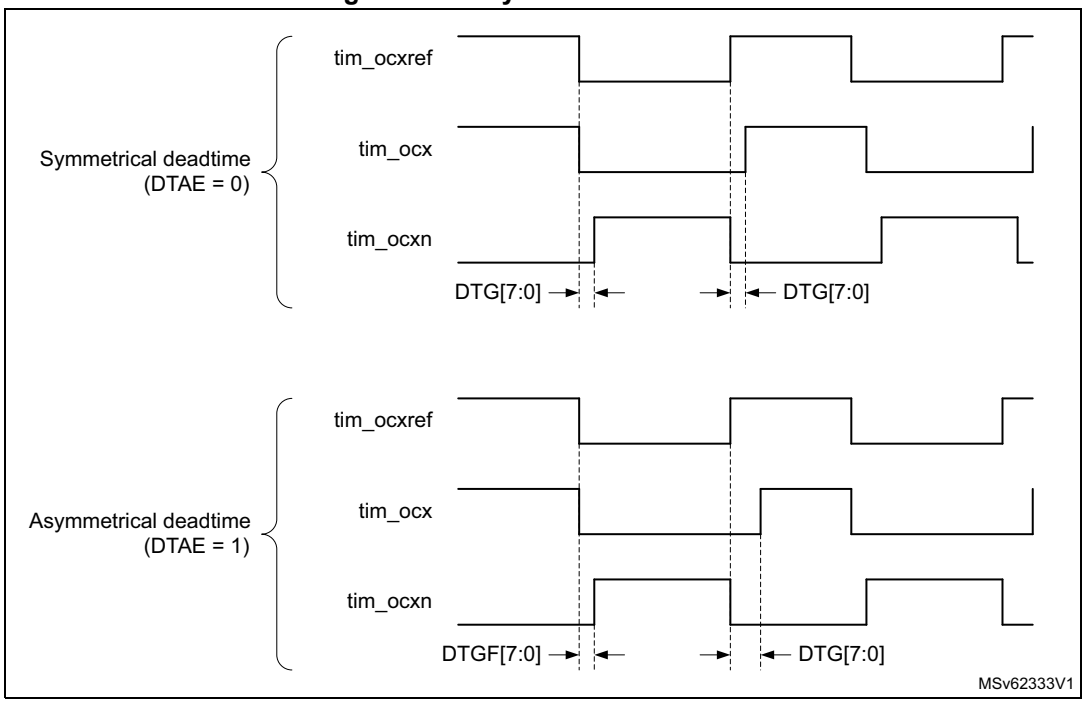


Figure 746. Dead-time waveforms with delay greater than the negative pulse.

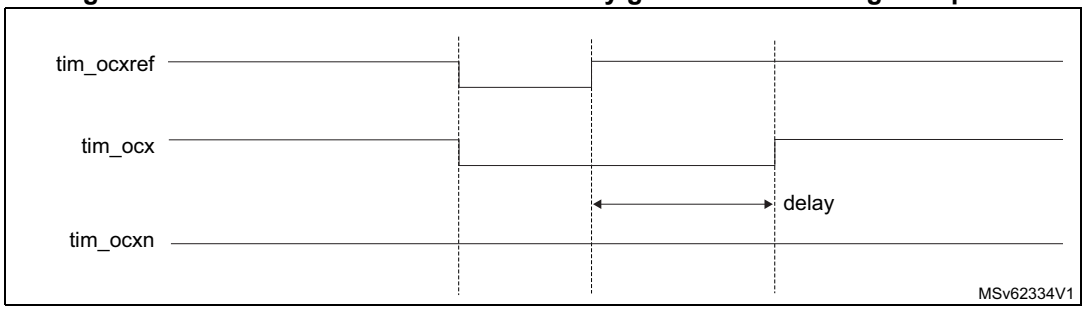
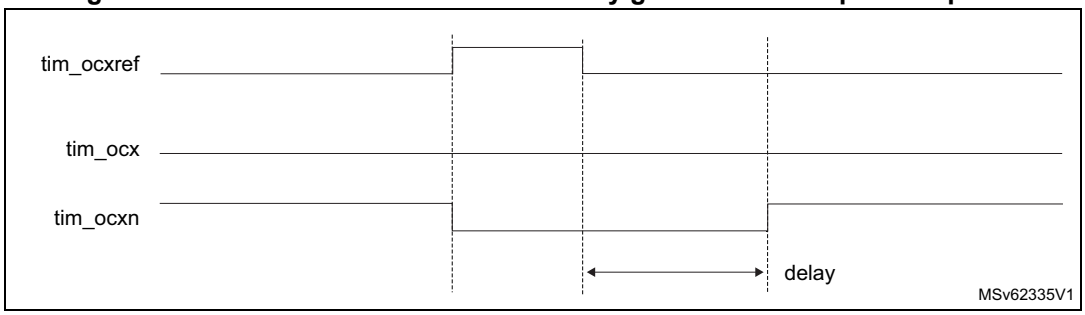


Figure 747. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIM_BDTR register. Refer to *Section 45.7.16: TIM break and dead-time register (TIM_BDTR) on page 2388* and *Section 45.8.14: TIM break and dead-time register (TIM_BDTR) on page 2420* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIM_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

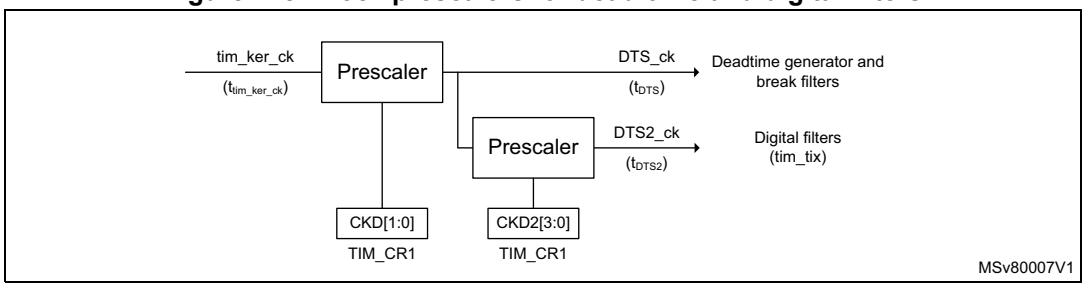
Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

45.4.16 Clock dividers for dead-time generators and digital filters

The dead-time generators and the digital filters (for tim_tix) are using a DTS_ck clock (with a t_{DTS} period), derived from the tim_ker_ck clock.

A prescaler can decrease the frequency to extend the deadtime period programmed with the CKD[1:0] bitfield in the TIM_CR1 register. A second prescaler can decrease the resulting DTS_ck clock to extend solely the digital filters range, with the CKD2[3:0] bitfield in the TIM_CR1 register.

Figure 748. Clock prescalers for dead-time and digital filters



45.4.17 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIM_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIM_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIM_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 531: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) on page 2384* and *Table 533: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 2416* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIM_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIM_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIM_BDTR register).
- Programmable enable bit (BKE bit in the TIM_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIM_BDTR register) to avoid spurious events.

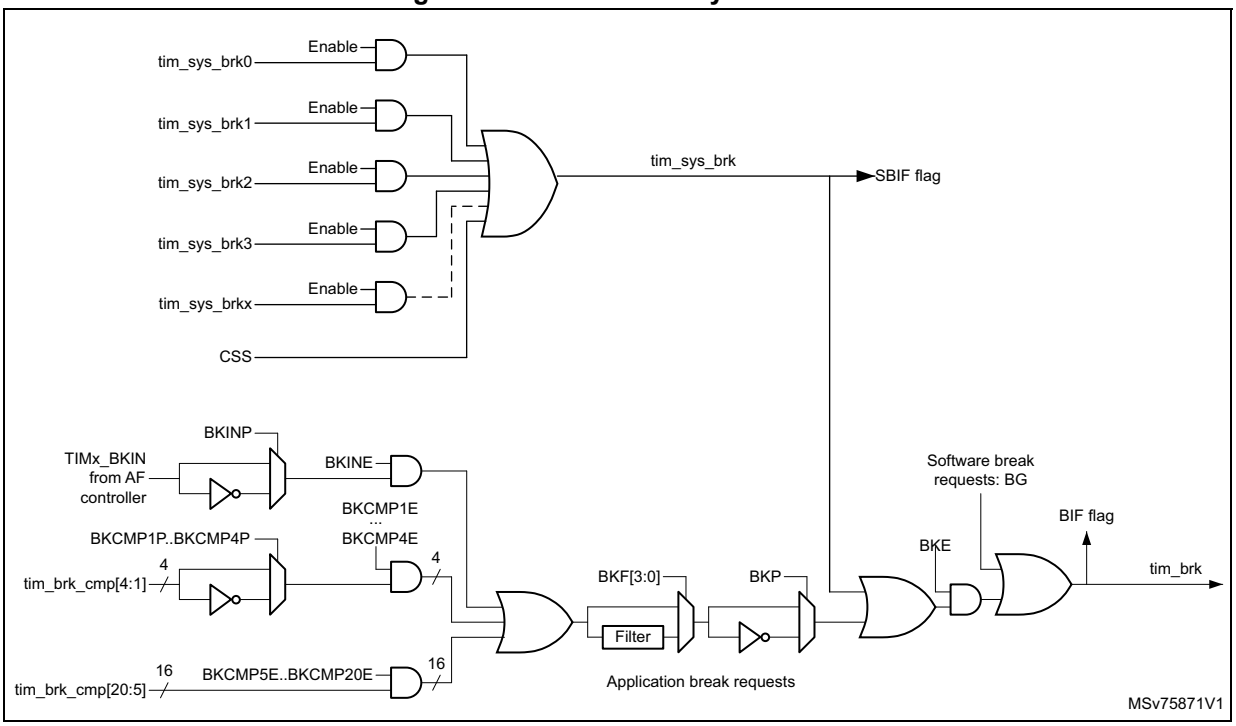
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIM_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).

Break events can also be generated by software using BG bit in the TIM_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 749* below.

Figure 749. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIM_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE,

the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF and BGF bits in the TIM_SR register) is set. An interrupt can be generated if the BIE bit in the TIM_DIER register is set. A DMA request can be sent if the BDE bit in the TIM_DIER register is set.
- If the AOE bit in the TIM_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIM_CR2 register.

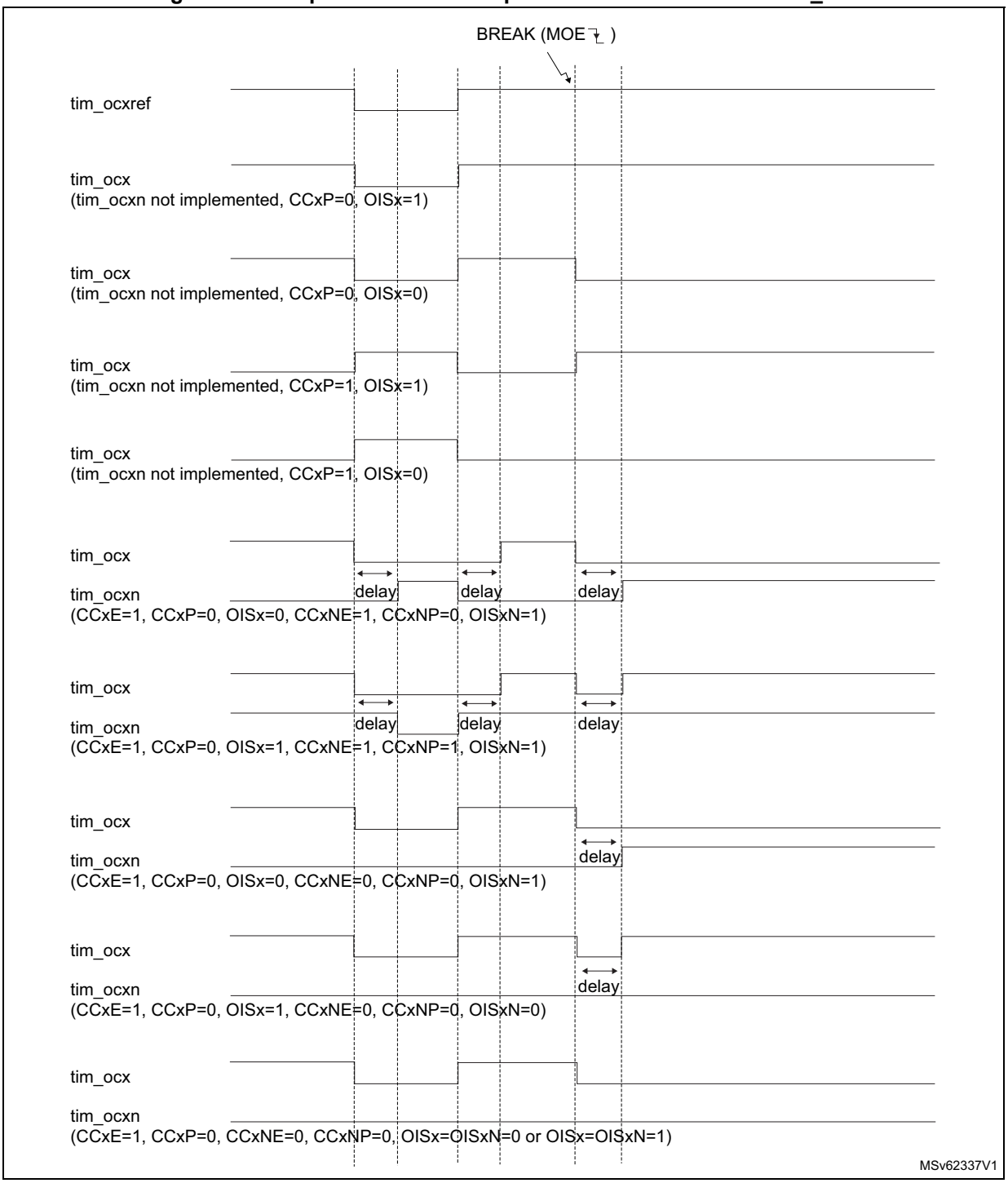
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIM_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIM_BDTR register. Refer to *Section 45.7.16: TIM break and dead-time register (TIM_BDTR)* and *Section 45.8.14: TIM break and dead-time register (TIM_BDTR)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 750* shows an example of behavior for the outputs in response to a break.

Figure 750. Output behavior in response to a break event on tim_brk



45.4.18 Delayed break

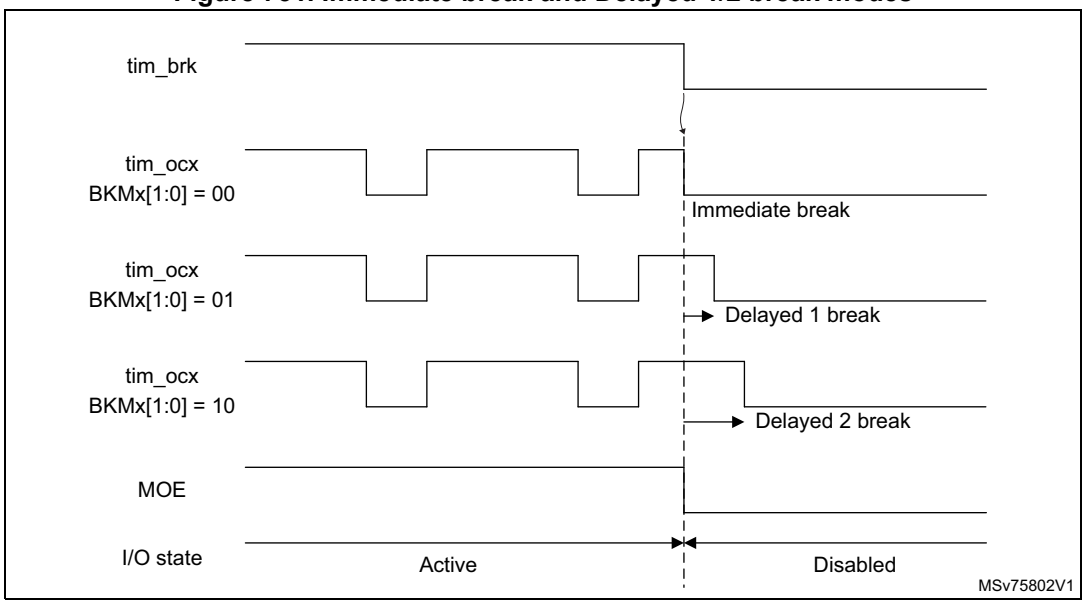
In Neutral-Point Clamped (NPC) multilevel converter topology, a break strategy with immediate outputs shut-down is not suitable, as it may damage the power switches with excessive voltage levels.

The delayed break features provide 2 options delaying the response to a break event, and implement a safe shut-down sequence.

BKMx[1:0] and BKMxN[1:0] bitfields in TIM_MPR1 provides 3 options for each complementary outputs (tim_oc1/tim_oc1n to tim_oc2/tim_oc2n), as show on *Figure 751* below:

- BKM[1:0] = 00: the break response is immediate
- BKM[1:0] = 01: Delayed 1 break. The break response is delayed with a delay defined in the DBK1[7:0] bitfield of the TIM_MPR2 register
- BKM[1:0] = 10: Delayed 2 break. The break response is delayed with a delay defined in the DBK2[7:0] bitfield of the TIM_MPR2 register

Figure 751. Immediate break and Delayed 1/2 break modes



The delayed break mode is triggered when:

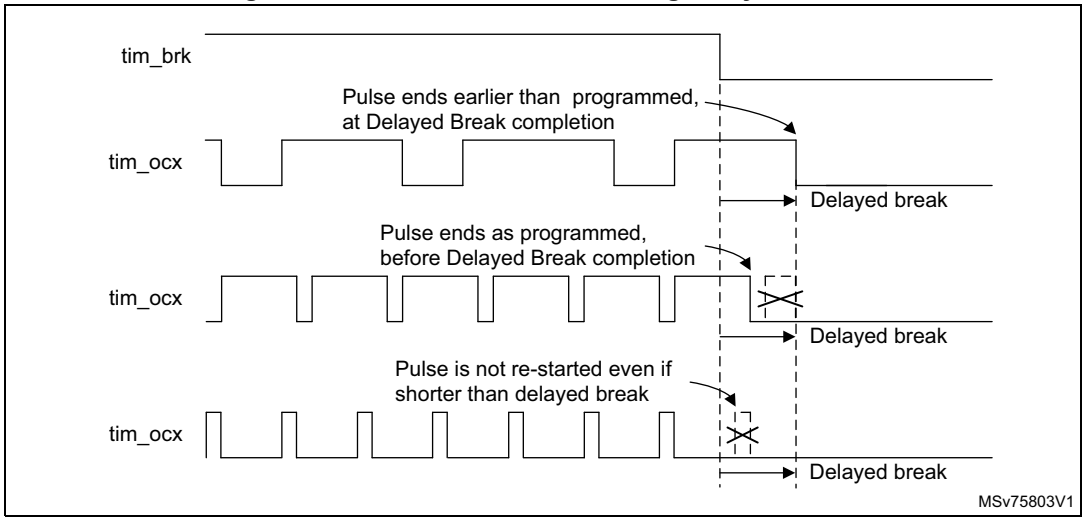
- A break event occurs
- The MOE bit is reset
- When the microcontroller enters debug mode with DBG_TIMx_STOP=1

When delayed break mode is triggered, the MOE bit is immediately cleared, and cannot be set during the delayed break duration. In bidirectional break mode, the break output is reset immediately to indicate the delayed break start to external components. Once the longest of the 2 delays is over, the deadtime generator is enabled to prevent any deadtime violation if OISx/OISxN bits are set.

If the tim_ocx/tim_ocxn signal is reset before the end of the delayed break (due to a compare match in PWM mode, typically), the signal is not maintained during the full delayed break period, to maintain the duty cycle, as shown on *Figure 752* below. Still, the Delayed

break circuitry guarantees that the tim_ocx/tim_ocxn signals cannot be set again once the break event occurred.

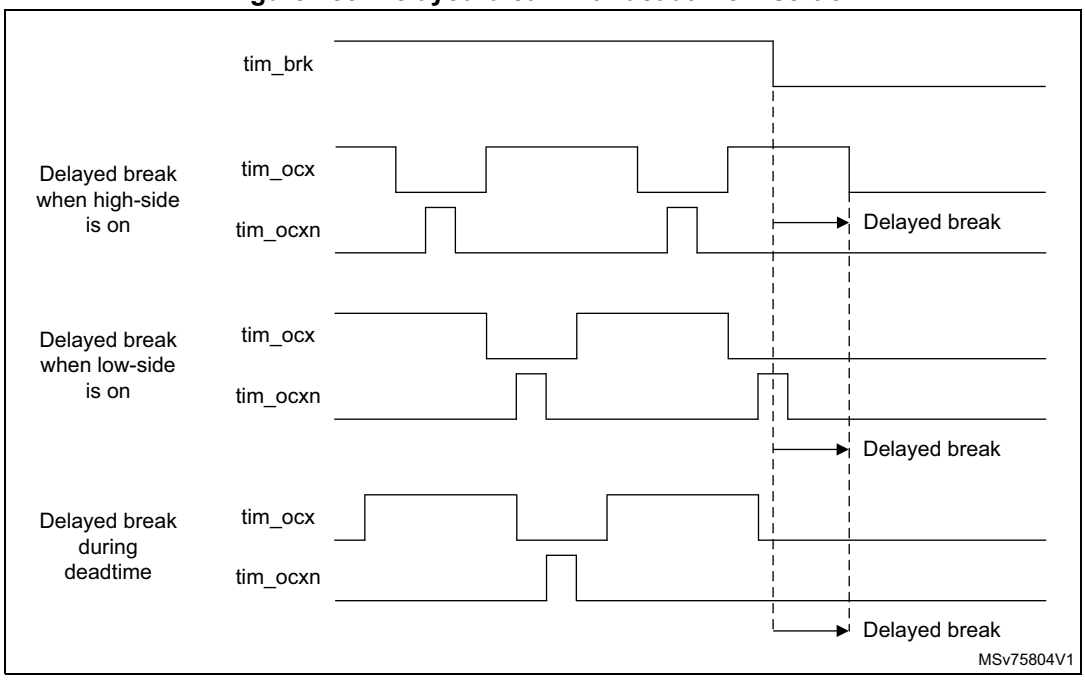
Figure 752. Pulse termination during delayed break



The *Figure 753* depicts the behavior when the deadtime generator is enabled, for the 3 cases:

- Delayed break triggered when tim_ocx is set
- Delayed break triggered when tim_ocxn is set
- Delayed break triggered during deadtime interval

Figure 753. Delayed break with deadtime insertion



The delay programming is identical to deadtime generator's, with 2 user-programmable durations in the TIM_MPR2 register: DBK1[7:0] bitfield for the Delayed 1 break and DBK2[7:0] bitfield for the Delayed 2 break.

DBK1[7:0] and DBK2[7:0] bitfields are preloaded: they can be changed during timer operation, and the new value is transferred to the active register when an update event occurs.

The bits coding for t_{DTG} selection are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBKx[6:0] if DBKx[7] = 0
- DBKx[5:0] if DBKx[7:6] = 10
- DBKx[4:0] if DBKx[7:5] = 110 or 111

The BKMx[1:0] and BKMxN[1:0] bits are static, they can only be modified when the counter is stopped (CEN=0 in TIM_CR1).

The content of the TIM_MPR1 and TIM_MPR2 register can be write protected using the LOCK[1:0] bits in the TIM_BDTR register.

45.4.19 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 754*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIM_BDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIM_BDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 525*).

Table 525. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

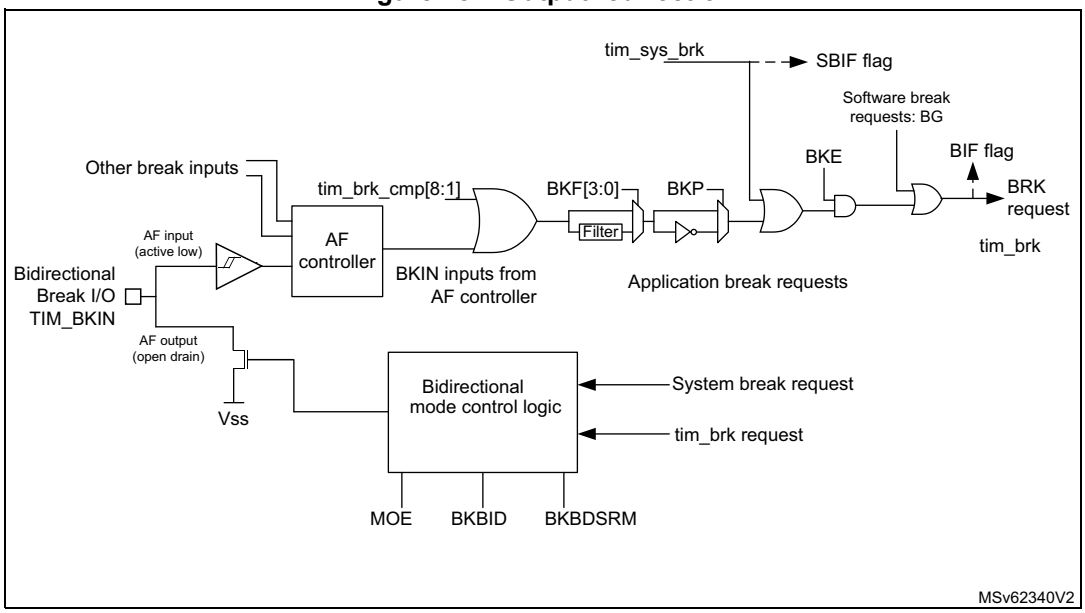
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 754. Output redirection



45.4.20 Output idle state override

It is possible to override the outputs states when they are disabled (ie MOE bit reset in TIM_BDTR). This feature can be used to apply specific patterns prior to start PWM operations, for instance to refresh bootstrap capacitors of power switch gate drivers.

The OOC control bit in the TIM_OOR register acts as following:

- OOC= 0: 0: tim_ocx and tim_ocxn outputs are in the default idle state
- OOC=1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits in the TIM_OOR register

The OOC bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared by hardware when the MOE bit is set, and the deadtime generator ensures there's no deadtime violation on complementary outputs.

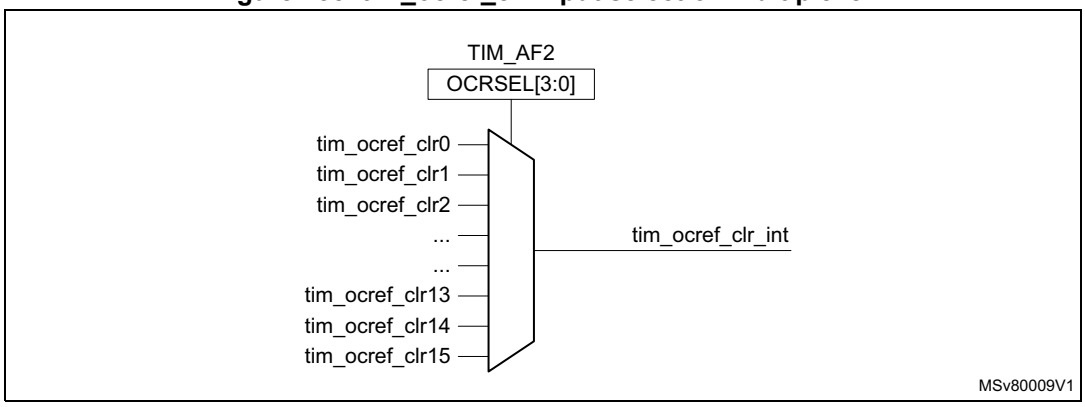
The OOSx/OOSxN bits in the TIM_OOR register cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

45.4.21 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIM_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It must not be used in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 755* below.

Figure 755. tim_ocref_clr input selection multiplexer



This feature is also supported in combined and asymmetric modes. In this case, it is mandatory to have the OCxCE bit enabled for the channel, the OCxCE bit of the complementary channel (for example OC2CE = X for oc2ref in *Figure 756*) is ignored in this case.

Figure 756 to *Figure 758* are describing the output waveforms for some use cases, when the ocref_clr_int signal is asserted.

Figure 756. Asymmetric PWM mode 1

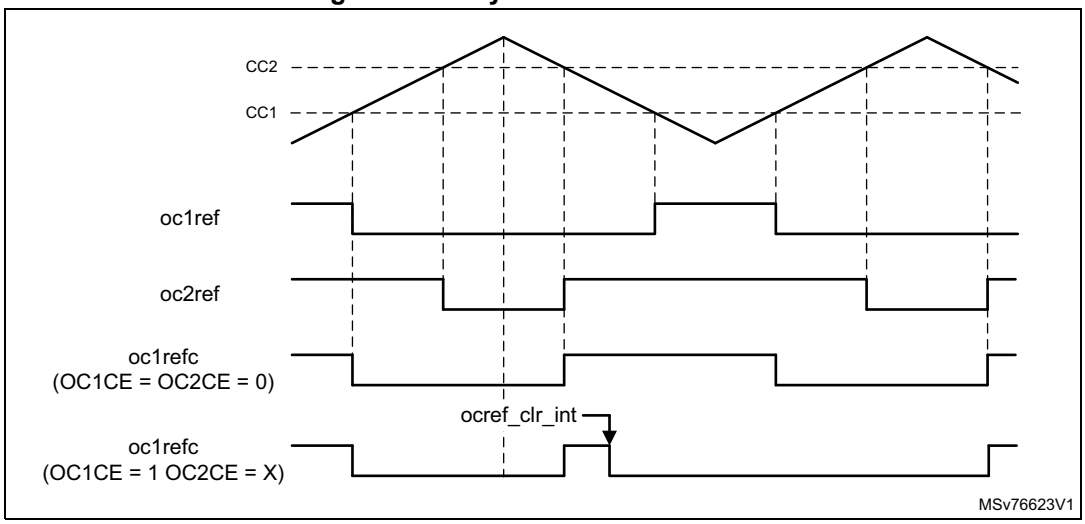


Figure 757. Asymmetric PWM mode 2

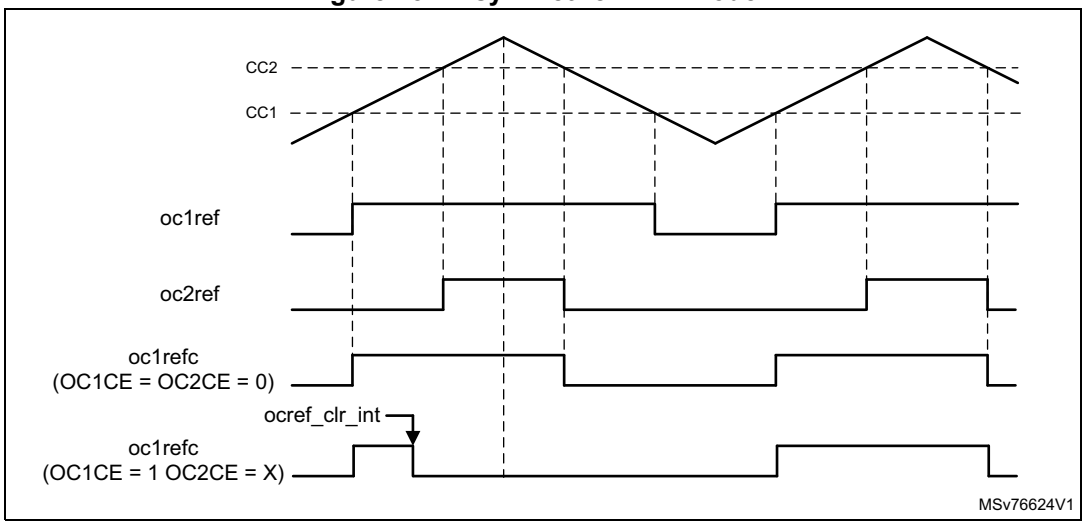
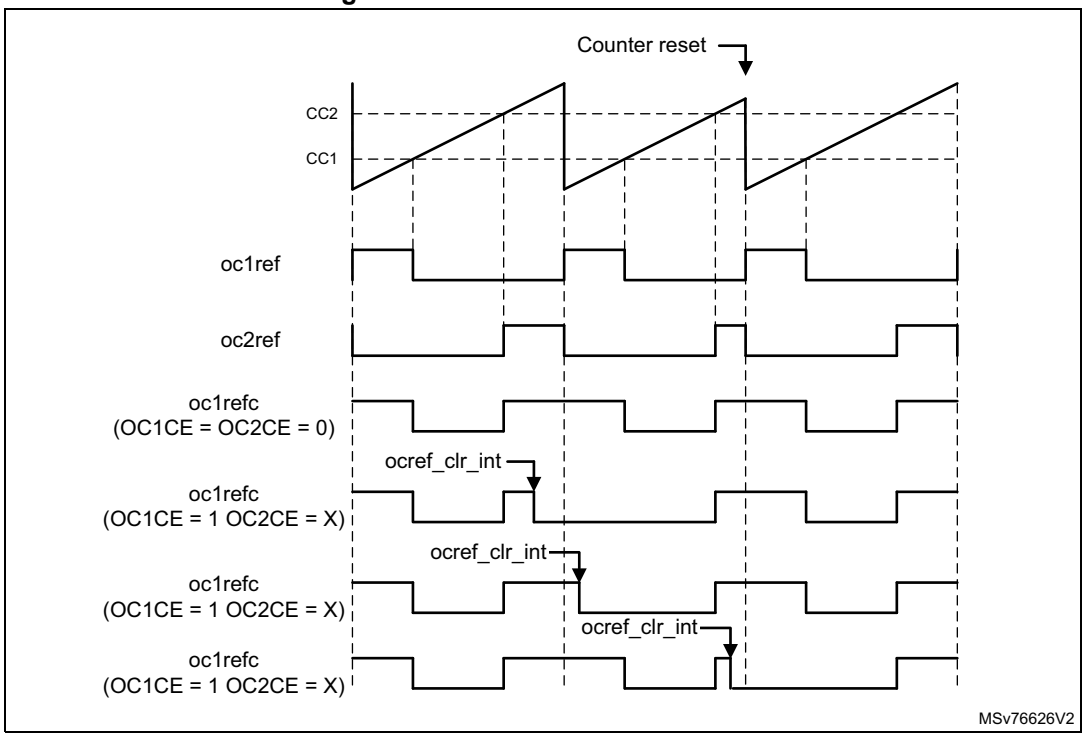


Figure 758. Combined PWM mode 1



Conditions of use

This feature is intended to support cycle-by-cycle current control loops, typically with a comparator shortening the PWM on-time when a current threshold is exceeded.

It cannot be used for the following modes, where the OCxCE bit must be cleared:

- PWM toggle mode (OCxM[4:0] = 0x03)
- One pulse mode with PWM active on match (OPM=1 and OCxM[4:0] = 0x06, 0x08, 0x0C, 0x0E)
- Combined PWM mode 1 in center aligned mode (CMS[1:0] = 1,2,3 and OCxM[4:0] = 0x0C)

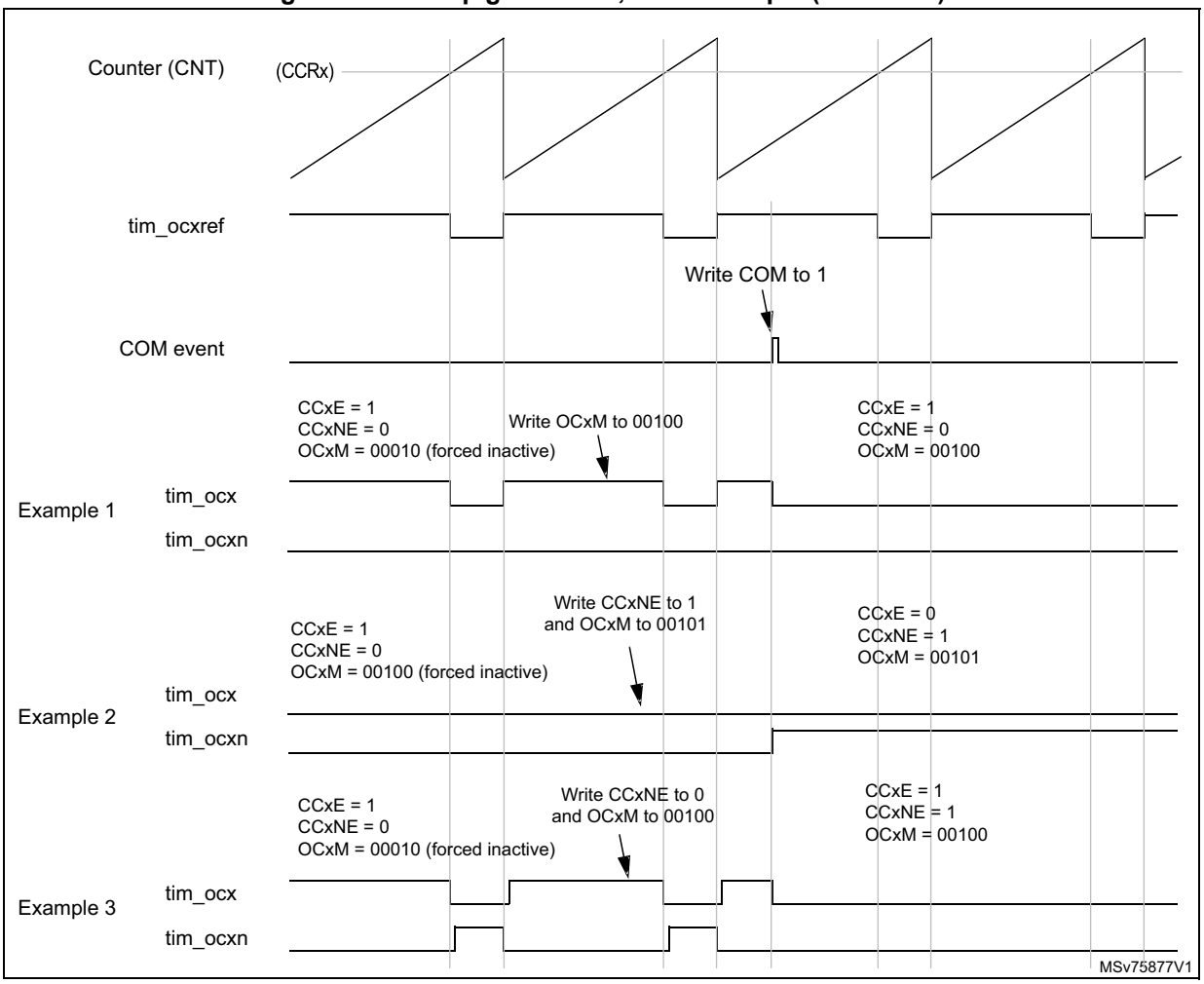
45.4.22 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIM_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIM_SR register), which can generate an interrupt (if the COMIE bit is set in the TIM_DIER register) or a DMA request (if the COMDE bit is set in the TIM_DIER register).

The *Figure 759* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 759. 6-step generation, COM example (OSSR = 1)



45.4.23 One-pulse mode

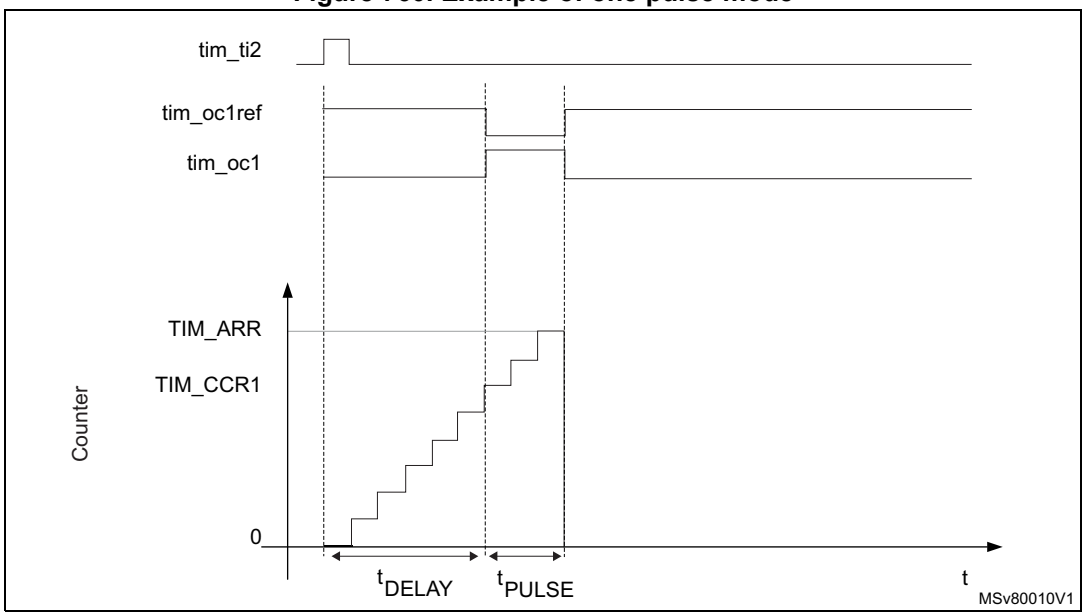
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIM_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 760. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIM_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIM_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIM_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIM_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 00110 in the TIM_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIM_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIM_ARR - TIM_CCR1).
- Let's say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 00111 in the TIM_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIM_CCMR1 register and ARPE in the TIM_CR1 register. In this case one has to write the compare value in the TIM_CCR1 register, the autoreload value in the TIM_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIM_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIM_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

45.4.24 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 45.4.23*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

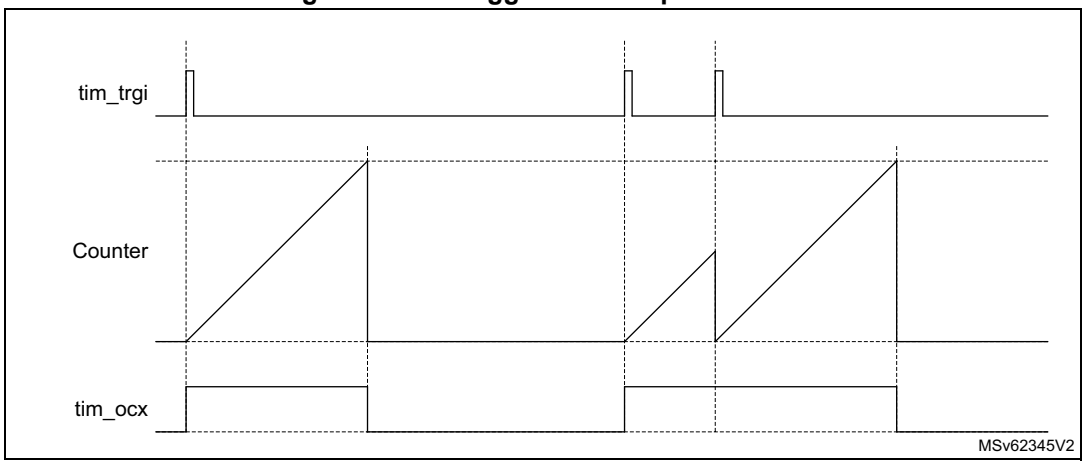
The timer must be in Slave mode, with the bits SMS[4:0] = 01000 (Combined reset + trigger mode) in the TIM_SMCR register, and the OCxM[4:0] bits set to 01000 or 01001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[4:0] and SMS[4:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIM_CR1.

Figure 761. Retriggerable one pulse mode



45.4.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 00001 in the TIM_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 00010 if it is counting on tim_ti2 edges only and SMS = 00011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIM_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 526*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIM_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIM_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIM_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIM_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 526. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	01110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	01111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	00001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	00010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	00011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 762 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIM_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIM_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIM_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIM_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 00011 (TIM_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM_CR1 register, Counter enabled).

Figure 762. Example of counter operation in encoder interface mode.

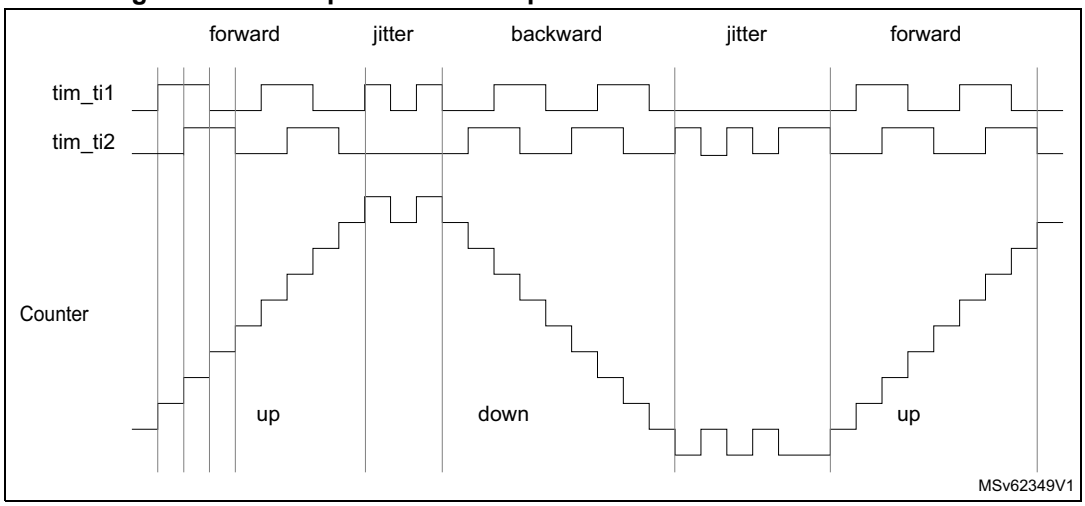


Figure 763 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 763. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

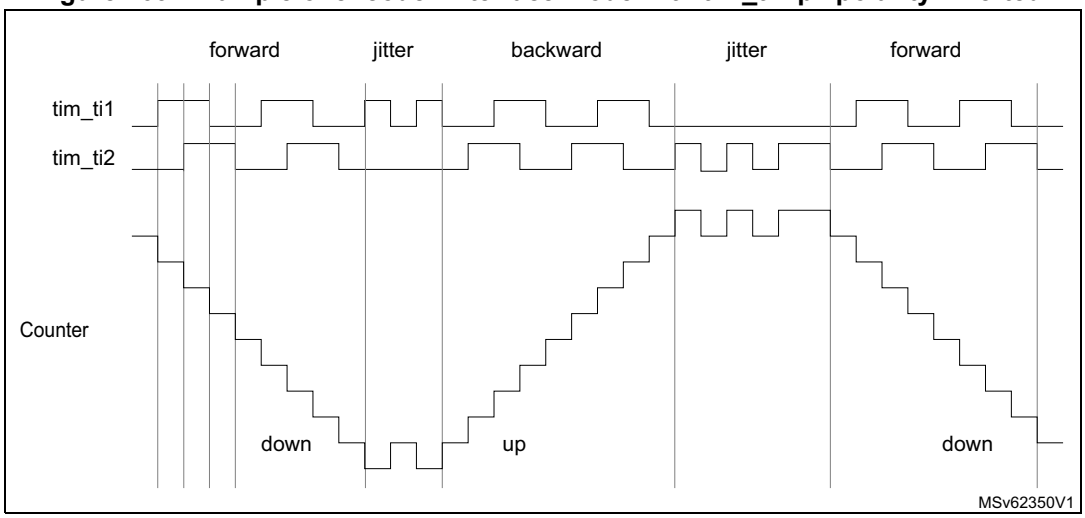
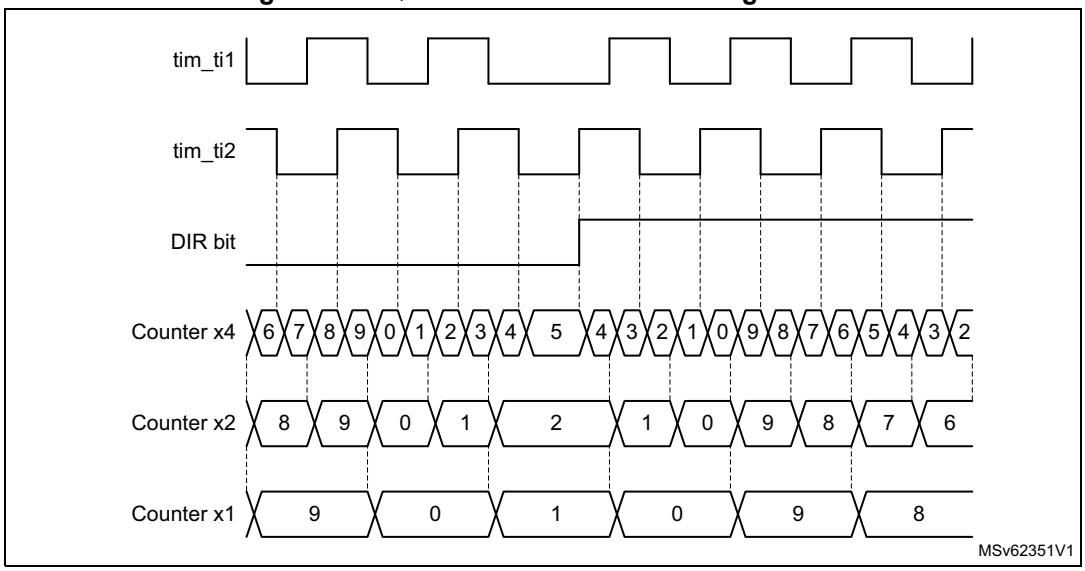


Figure 764 shows the timer counter value during a speed reversal, for various counting modes.

Figure 764. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIM_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

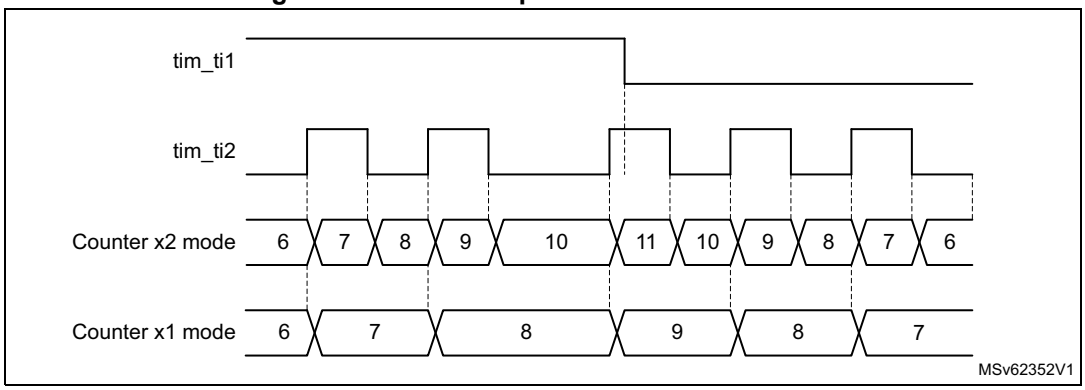
In the clock plus direction mode shown on *Figure 765*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 01011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 765. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 766*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[4:0] bitfield in the TIM_SMCR register, as following:

- 01100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 766*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 767*).
- 01101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 766*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 767*).

Figure 766. Directional clock encoder mode (CC1P = CC2P = 0)

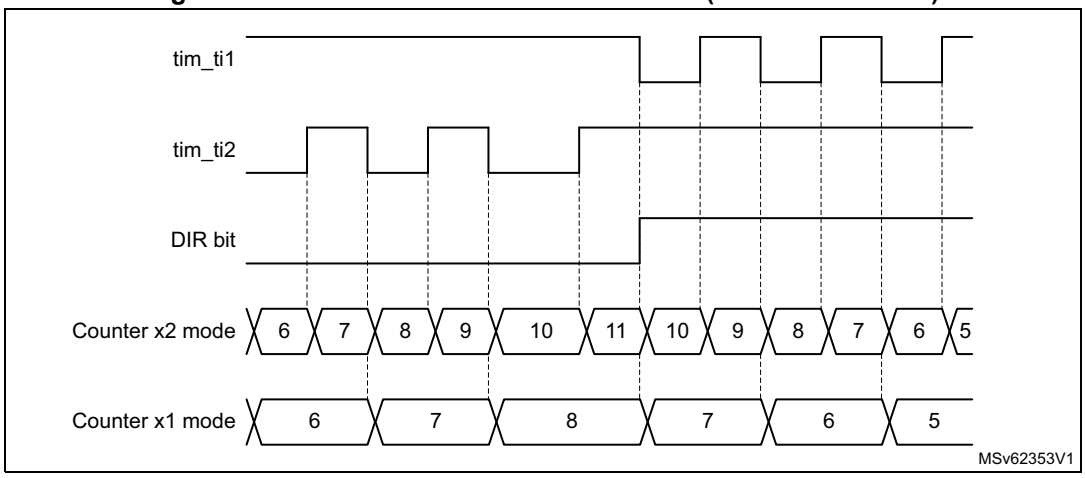


Figure 767. Directional clock encoder mode (CC1P = CC2P = 1)

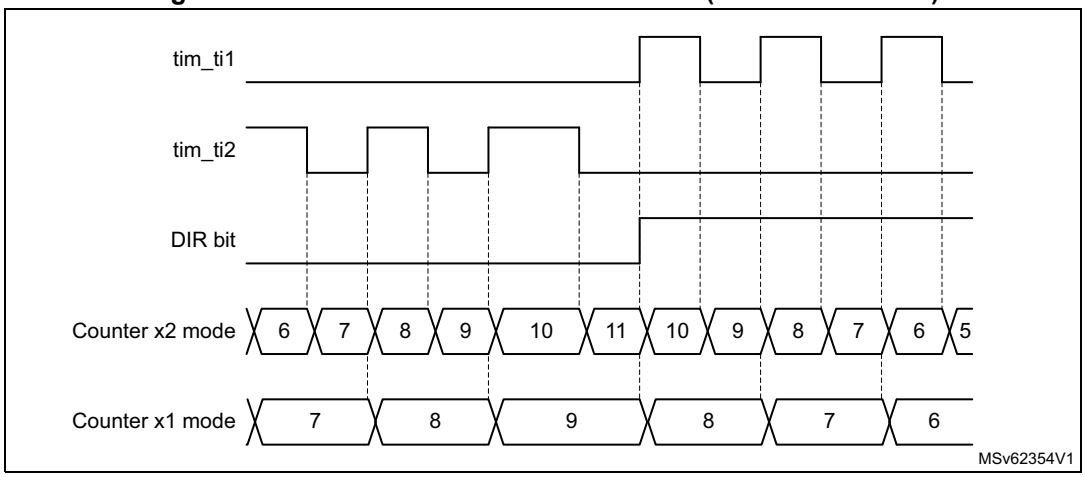


Table 527 here-below details how the directional clock mode operates, for any input transition.

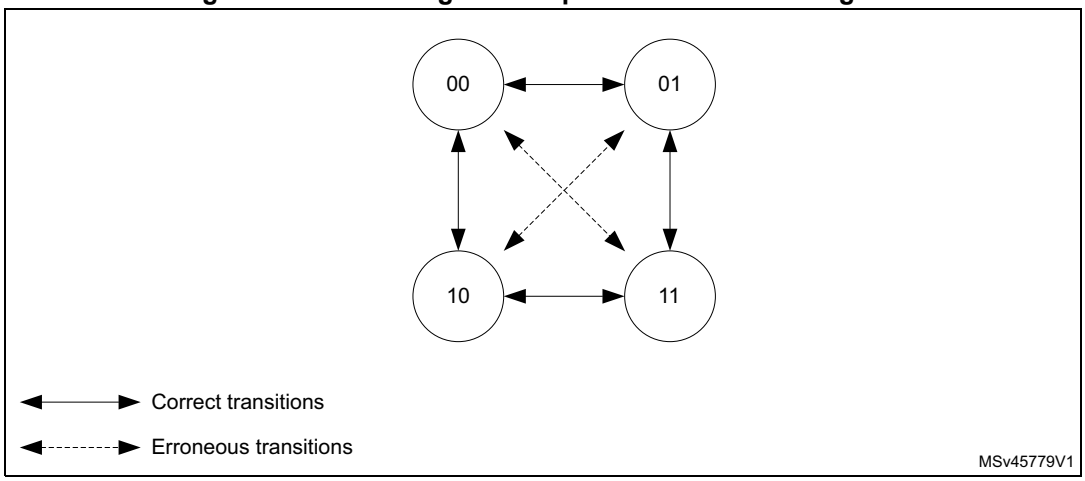
Table 527. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[4:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	01100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	01100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	01101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	01101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 768.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIM_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIM_DIER register.

Figure 768. State diagram for quadrature encoded signals



Functional encoder interrupts

The following interrupt is also available in encoder mode

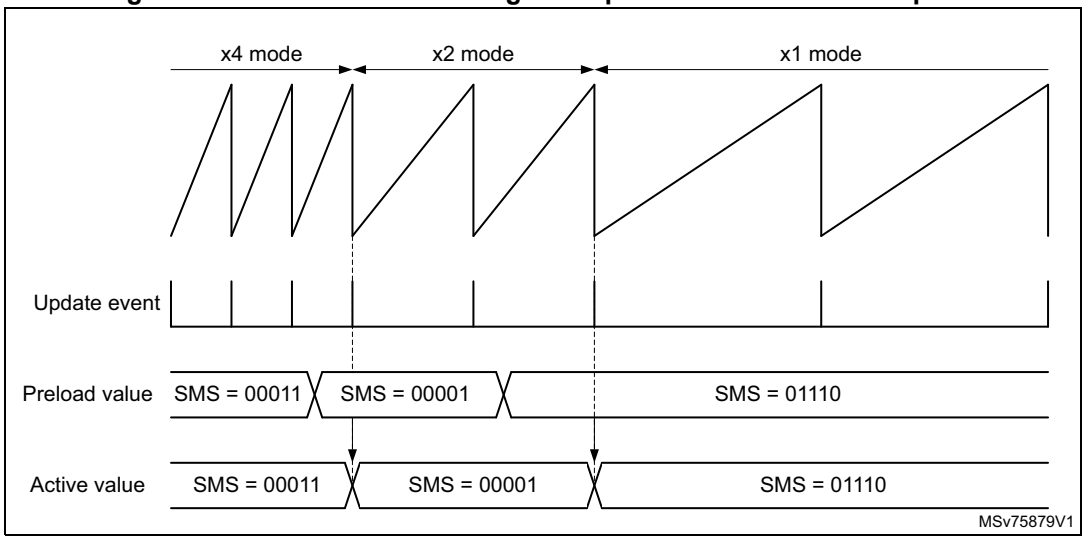
- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIM_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIM_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIM_DIER register.

Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 769*.

For this purpose, the SMS[4:0] bit can be preloaded. The transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.

Figure 769. Encoder mode change with preload transferred on update



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIM_CR2 register.It is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 10101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

45.4.26 Encoder with built-in debouncer (TIM15 only)

This mode is intended to support noisy digital potentiometers and knobs, providing quadrature encoded signals with potentially long bouncing periods after each transition.

The operating principle consists of working on the first valid transition and discard any other until a second valid transition is detected on the alternate encoder signal.

The *Figure 770* below represents the x2 operating mode. It must be noticed that a transition on input 1 is ignored during direction reversal, if there's no edge on input 2.

The *Figure 771* below represents the x4 operating mode. In this case, a transition is always ignored during a direction reversal or when the encoder dithers between 2 positions.

These 2 modes are enabled by writing SMS[4:0] = 10000 or 10001 in TIM_SMCR register.

The encoder with built-in debouncer modes are not suitable for tracking a position or an angle in motor control applications. The transitions ignored during direction reversal will lead to a position/angle error that will build-up over time.

Figure 770. Encoder with built-in debouncer, x2 mode

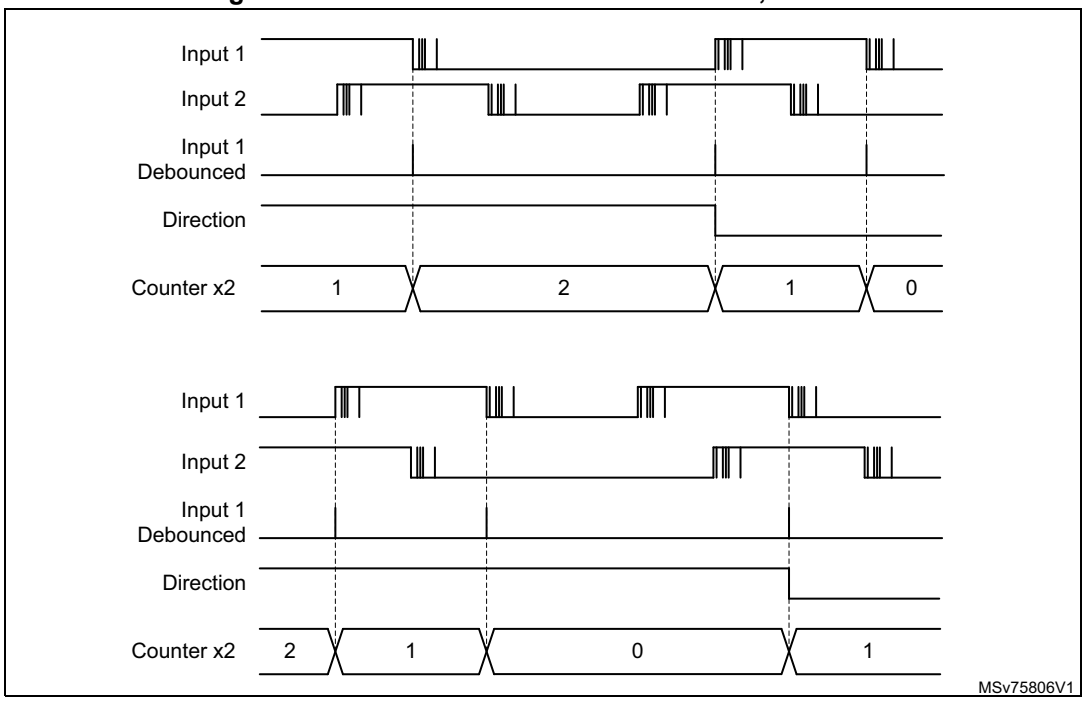
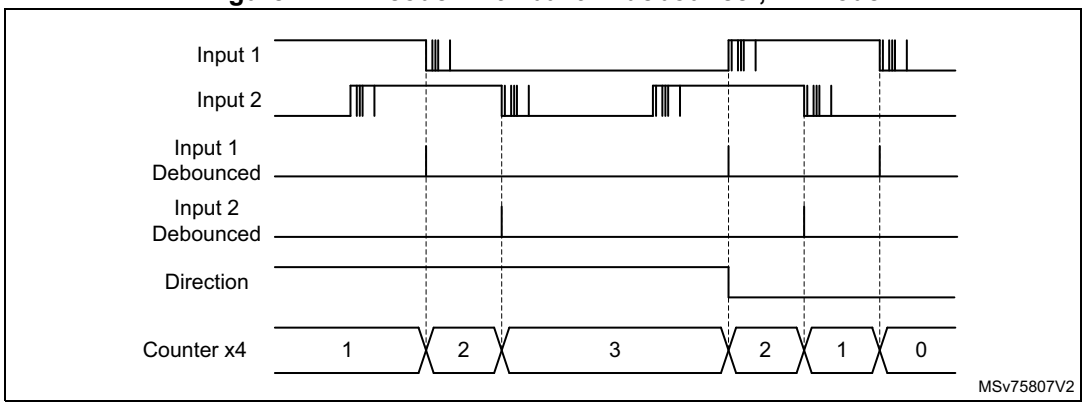


Figure 771. Encoder with built-in debouncer, x4 mode



45.4.27 Clock drift measurement

It is possible to measure the drift between two clock sources using the directional clock encoder mode. This is useful for audio applications typically, when one need to compute the drift between two streams.

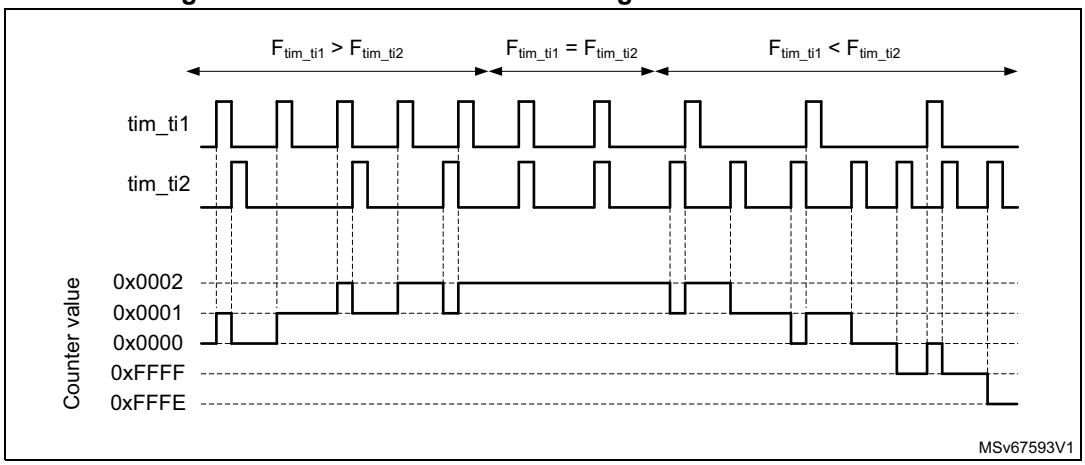
To do so, a signal representing the rate of a first audio stream must be connected to tim_ti1 and a signal representing the rate of a second audio stream must be connected to tim_ti2. The signals representing the audio rates are generally the SAIx_FS_A, SAIx_FS_B, SPIx_WS or spdifrx_frame_sync (this list of signals is provided as example and depends on the audio peripheral availability for a given product line, as detailed on the product datasheet).

In this case, the clocks are present on both tim_ti1 and tim2 simultaneously, as shown on *Figure 772*, and the counter value indicates the drift.

Let's define F_{tim_ti1} and F_{tim_ti2} the frequencies of the signals on tim_ti1 and tim_ti2. The counter value changes as following:

- If $F_{tim_ti1} > F_{tim_ti2}$ the counter value increases
- If $F_{tim_ti1} < F_{tim_ti2}$ the counter value decreases
- If $F_{tim_ti1} = F_{tim_ti2}$ the counter value dithers between two consecutive values, from one clock edge to the other. If the two clocks are perfectly synchronized (clock edges occuring simultaneously on the two inputs), the counter value does not change (as shown on *Figure 772* below).

Figure 772. Clock drift measure using directional clock mode



45.4.28 UIF bit remapping

The IUFREMAP bit in the TIM_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIM_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

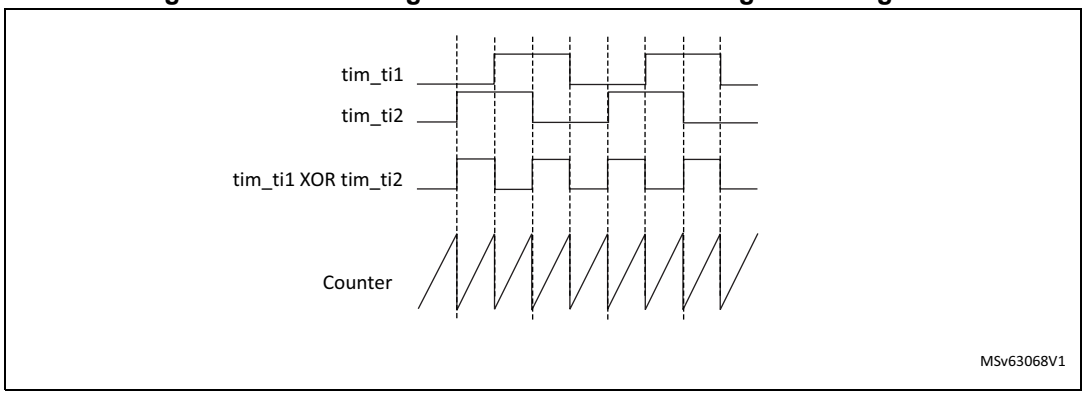
There is no latency between the assertions of the UIF and UIFCPY flags.

45.4.29 Timer input XOR function (TIM15 only)

The TI1S bit in the TIM_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 773*.

Figure 773. Measuring time interval between edges on 2 signals



The XORPS control bit selects the position of the XOR gate, as shown in the *Figure 774* and *Figure 775* below, when the TI1S bit is set.

Figure 774. XOR gate connection with TI1S = 1 and XORPS = 0

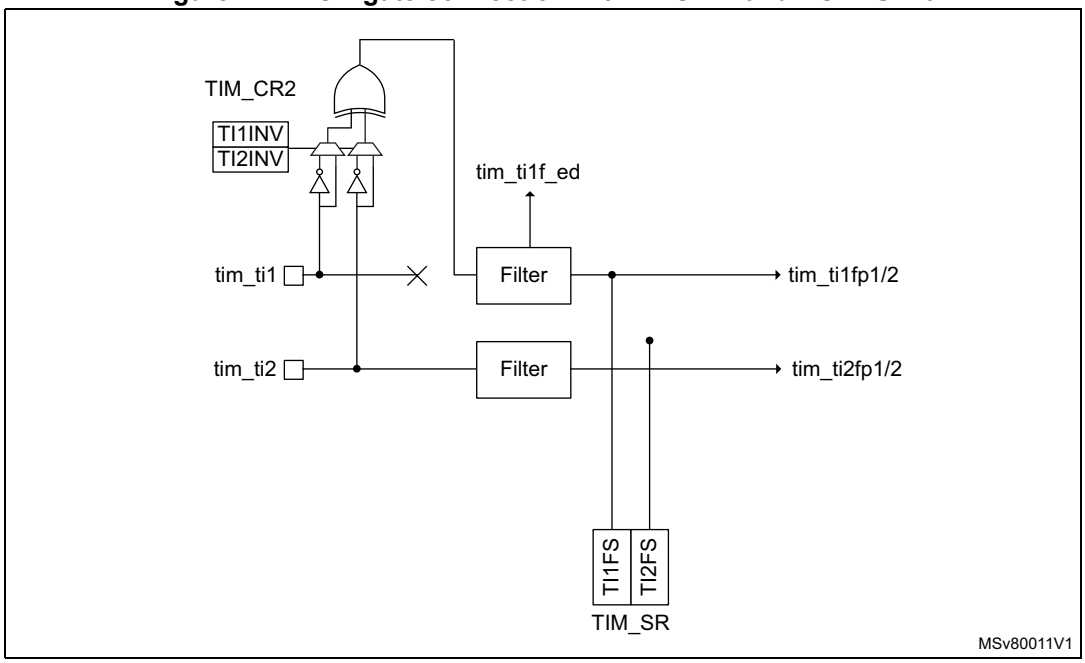
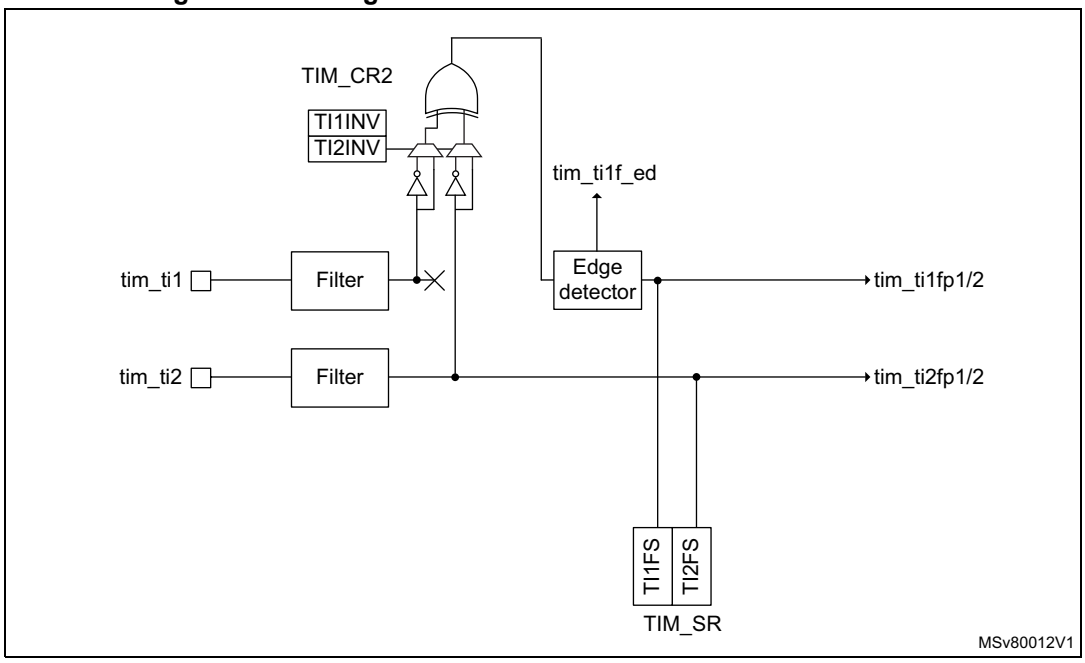


Figure 775. XOR gate connection with TI1S = 1 and XORPS = 1



Two control bits TI1INV and TI2INV in the TIM_CR2 register can invert the tim_in1 and tim_in2 signals prior entering the XOR gate.

The status of the tim_in1 and tim_in2 signals can be software polled using the TI1FS and TI2FS bits in the TIM_SR register.

45.4.30 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIM_ARR, TIM_CCRx) are updated.

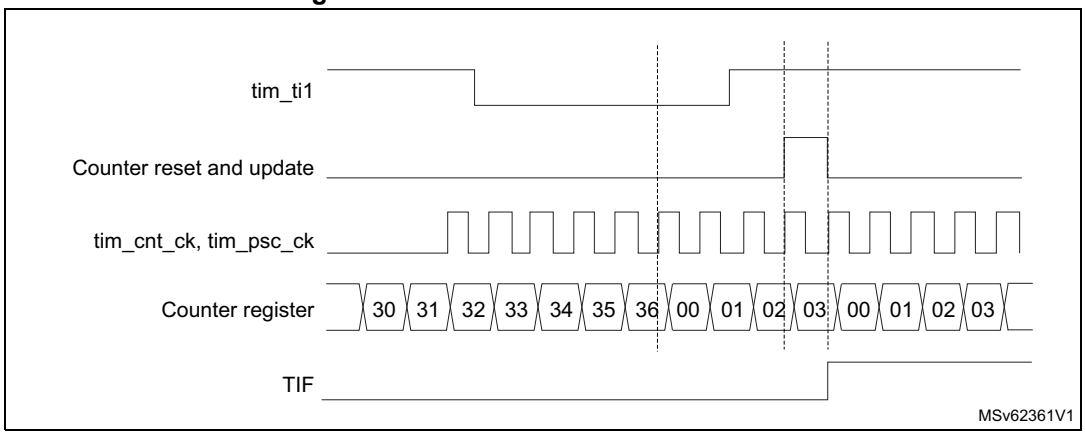
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIM_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in the TIM_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 00100 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIM_DIER register).

The following figure shows this behavior when the autoreload register TIM_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 776. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

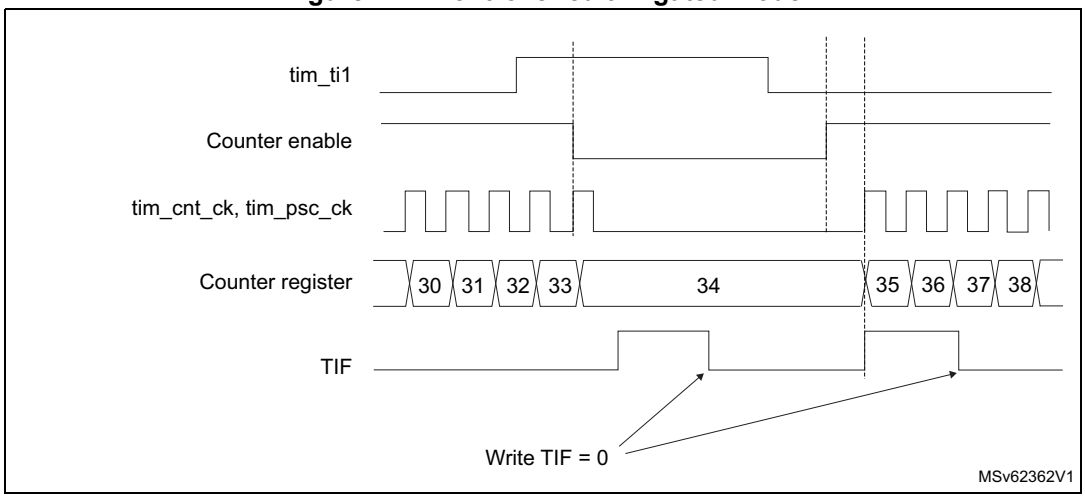
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIM_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in the TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 00101 in TIM_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIM_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIM_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 777. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

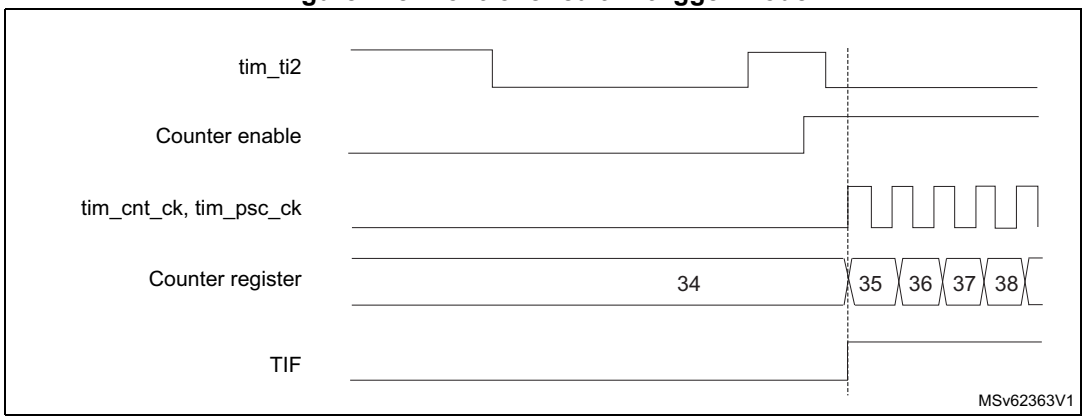
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, no need for any filter, so IC2F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIM_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in the TIM_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 00110 in the TIM_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in the TIM_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 778. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[4:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIM_SMCR register. The trigger for the transfer from SMS[4:0] preload to active value is the update event (UEV) occurring when the counter overflows.

45.4.31 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

45.4.32 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

45.4.33 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 42.4.27: Timer synchronization* for details.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

45.4.34 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

45.4.35 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

45.4.36 ADC synchronization (TIM15 only)

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIM_CR2 register.

This feature is useful when the timers and the ADCs are operating with semi-synchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the 2 requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times (\text{TIM_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIM_PSC} + 1) \times \text{TIM_CCRx} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

If none of the 2 above requirement are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

45.4.37 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIM_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIM_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIM_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIM_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.

Example:

00000: TIM_CR1

00001: TIM_CR2

00010: TIM_SMCR

The DBSS[3:0] bits in the TIM_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 45.7.23: TIM output override register (TIM_OOR)* and *Section 45.8.21: TIM output override register (TIM_OOR)* for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 2, DBA = 0xE and DBSS[3:0] = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

45.4.38 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA request, as shown in *Table 528*.

Table 528. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

45.4.39 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

45.5 TIM15/TIM16/TIM17 low-power modes

Table 529. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

45.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 530*.

Table 530. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
TIM_TRG_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
	System Break	SBIF		write 0 in SBIF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

1. Available for TIM15 only.

45.7 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

45.7.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1).

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

45.7.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw		rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TI2INV**: tim_ti2 signal inversion on XOR gate input

0: tim_ti2 non-inverted

1: tim_ti2 inverted

Bit 29 **TI1INV**: tim_ti1 signal inversion on XOR gate input

0: tim_ti1 non-inverted

1: tim_ti1 inverted

Bit 28 **ADSYNC**: ADC synchronization
0: The timer operates independently from the ADC
1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 45.4.36* for requirements.
The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bit 27 **XORPS**: XOR gate position selection
This bit defines the position of the XOR gate with respect to input filters.
0: XOR gate placed before TI1 filter
1: XOR gate placed after TI1 and TI2 filters

Bit 26 Reserved, must be kept at reset value.

Bits 24:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)
0: tim_oc2 = 0 when MOE = 0
1: tim_oc2 = 1 when MOE = 0
Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 after a dead-time when MOE = 0
1: tim_oc1 = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIM_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[4:0] values: 00001, 00010, 00011, 01010, 01011, 01100, 01101, 01110, 01111. Any other SMS[4:0] code is not allowed and may lead to unexpected behavior.

Others: reserved.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

45.7.3 TIM slave mode control register (TIM_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	SMS[4:3]	
							rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE:** SMS preload enable

This bit selects whether the SMS[4:0] bitfield is preloaded.

0: SMS[4:0] bitfield is not preloaded

1: SMS[4:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if the user wants to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
10100: Internal trigger 16 (tim_itr16)
10101: Internal trigger 17 (tim_itr17)
10110: Internal trigger 18 (tim_itr18)
10111: Internal trigger 19 (tim_itr19)
11000: Internal trigger 20 (tim_itr20)
11001: Internal trigger 21 (tim_itr21)
11010: Internal trigger 22 (tim_itr22)
11011: Internal trigger 23 (tim_itr23)
11100: Internal trigger 24 (tim_itr24)
11101: Internal trigger 25 (tim_itr25)
11110: Internal trigger 26 (tim_itr26)
11111: Internal trigger 27 (tim_itr27)
Others: Reserved

See *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 17:16, 2:0 **SMS[4:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIM_SMCR for tim_etr_in and CCxP/CCxNP bits in TIM_CCER register for tim_ti1fp1 and tim_ti2fp2).

00000:Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

00001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

00010:Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

00011:Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

00100:Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

00101:Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

00110:Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

00111:External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

01000:Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

01001:Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

01010:Encoder mode: Clock plus direction, x2 mode.

01011:Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

01100:Encoder mode: Directional Clock, x2 mode.

01101:Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

01110:Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

01111:Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

10000:Quadrature encoder with built-in debouncer: x2 mode.

10001:Quadrature encoder with built-in debouncer: x4 mode.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

45.7.4 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.
								rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 Reserved, must be kept at reset value.

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

45.7.5 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	Res.	DIRF	Res.	Res.	Res.	Res.	Res.
		r	r				rc_w0	rc_w0		rc_w0					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BGF	SBIF	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	rc_w0	rc_w0			rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TI2FS**: tim_ti2f status
This flag indicates the value of the tim_ti2f signal (after the digital filtering stage), for polling purpose.
0: tim_ti2f signal is low
1: tim_ti2f signal is high

Bit 28 **TI1FS**: tim_ti1f status
This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.
0: tim_ti1f signal is low
1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag
This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.
0: No Update Interrupt Overrun
1: Update Interrupt Overrun

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 Reserved, must be kept at reset value.

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIM_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **BGF**: Break generation flag
This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.
0: No break event occurred.
1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected
1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 45.7.3: TIM slave mode control register (TIM_SMCR)*), if URS = 0 and UDIS = 0 in the TIM_CR1 register.

45.7.6 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1 A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

45.7.7 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

45.7.8 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
						rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:26 Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable
Refer to OC1CE description on bit 7.

Bits 25:24, 14:12 **OC2M[4:0]**: Output compare 2 mode
Refer to OC1M description on bits 6:4.

Bit 11 **OC2PE**: Output compare 2 preload enable
Refer to OC1PE description on bit 3.

Bit 10 **OC2FE**: Output compare 2 fast enable
Refer to OC1FE description on bit 2.

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.
10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input.
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.
Note: This mode is not available for all PWM modes. Refer to Section 45.4.21: Clearing the tim_ocxref signal on an external event for the list of forbidden configurations.

Bits 17:16, 6:4 **OC1M[4:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - Channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive.

00111: PWM mode 2 - Channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active.

01000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

01001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

01010: Reserved

01011: Reserved

01100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

01101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

01110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

01111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

45.7.9 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 531* for details.

Table 531. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0 tim_ocxn = 0	
		1	0	0	Off-State (output enabled with inactive state) tim_ocx=CCxP, tim_ocxn=CCxNP	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP Then if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

45.7.10 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM_SR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

45.7.11 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

45.7.12 TIM autoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 45.4.3: Time-base unit on page 2294* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

45.7.13 TIM repetition counter register (TIM_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

45.7.14 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIM_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

45.7.15 TIM capture/compare register 2 (TIM_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIM_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

45.7.16 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 45.7.9: TIM capture/compare enable register (TIM_CCER) on page 2382*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1: Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 45.7.9: TIM capture/compare enable register (TIM_CCER) on page 2382*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 45.7.9: TIM capture/compare enable register (TIM_CCER) on page 2382*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal = 0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. tim_ocx/tim_ocxn enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

45.7.17 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable
0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register
1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup
This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.
DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.
DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.
DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.
DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps
Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

45.7.18 TIM input selection register (TIM_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input
0000: TIM15_CH2 input (tim_ti2_in0)
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIM15_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

45.7.19 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable

This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp20 input disabled

1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable

This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp13 input disabled

1: tim_brk_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable

This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp12 input disabled

1: tim_brk_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable

This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp11 input disabled

1: tim_brk_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable

This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp10 input disabled

1: tim_brk_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable

This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp9 input disabled

1: tim_brk_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

45.7.20 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

0000: tim_ocref_clr0

0001: tim_ocref_clr1

...

1111: tim_ocref_clr15

Refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

45.7.21 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **BK2M[1:0]**: Break mode for tim_oc2 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

45.7.22 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for Tdtg selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK2[6:0] if DBK2[7] = 0
- DBK2[5:0] if DBK2[7:6] = 10
- DBK2[4:0] if DBK2[7:5] = 110 or 111

DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.

It is partially preloaded and can be updated during timer operation (new value taken into account after an update event), at the exception of the bits coding for Tdtg selection, which are not preloaded.

It is thus recommended to update only the least significant bits to have the expected duration value:

- DBK1[6:0] if DBK1[7] = 0
- DBK1[5:0] if DBK1[7:6] = 10
- DBK1[4:0] if DBK1[7:5] = 110 or 111

DBK1[7:5]=0xx => Delay1=DBK1[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK1[7:5]=10x => Delay1=(64+DBK1[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK1[7:5]=110 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK1[7:5]=111 => Delay1=(32+DBK1[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

45.7.23 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS2	OOS1N	OOS1
			rw										rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control

This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.

0: tim_ocx and tim_ocxn outputs are in the default idle state

1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits

Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:3 Reserved, must be kept at reset value.

Bit 2 **OOS2**: Output override state for output tim_oc2

This bit defines the state for output tim_oc2 when OOC = 1

0: tim_oc2 = 0

1: tim_oc2 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **OOS1N**: Output override state for output tim_oc1n

This bit defines the state for output tim_oc1n when OOC = 1

0: tim_oc1n = 0

1: tim_oc1n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1

This bit defines the state for output tim_oc1 when OOC = 1

0: tim_oc1 = 0

1: tim_oc1 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

45.7.24 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).
0000: Reserved
0001: Update
0010: CC1
0011: CC2
0110: COM
0111: Trigger
Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length
This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).
00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11111: 32 transfers

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address
This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.
Example:
00000: TIM_CR1
00001: TIM_CR2
00010: TIM_SMCR
...

45.7.25 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

45.7.26 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 532. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM_CR2**	Res.	TI2INV	TI1INV	ADSYNC	XORPS	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value		0	0	0	0		0															0	0	0	0	0	0	0	0	0		0
0x08	**TIM_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	SMS[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value								0			0	0			0	0									0	0	0	0		0	0	0
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	DIRIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value										0	0							0	0			0	0	0	0	0	0			0	0	0
0x10	**TIM_SR**	Res.	Res.	TI2FS	TI1FS	Res.	Res.	Res.	UIOVRF	TERRF	Res.	DIRF	Res.	Res.	Res.	Res.	Res.	Res.	BGF	SBIF	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value			0	0				0	0		0							0	0			0	0		0	0	0			0	0	0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value							0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 532. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIM_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x60	**TIM_AF1**	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 532. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x64	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x68 - 0x08C	Reserved	Res.																															
0x090	**TIM_MPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2M[1:0]		BK1NM[1:0]		BK1M[1:0]	
	Reset value																											0	0	0	0	0	0
0x094	**TIM_MPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x98 - 0x09C	Reserved	Res.																															
0x0A0	**TIM_OOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS2	OOS1N	OOS1
	Reset value																				0										0	0	0
0xA4 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

45.8 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

45.8.1 TIM control register 1 (TIM_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CKD2[3:0]**: Clock division 2

This bit field indicates the division between sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_etr_in, tim_tix).

0000: No Division
0001: $t_{DTS2} = 4*t_{DTS}$
0010: $t_{DTS2} = 16*t_{DTS}$
0011: $t_{DTS2} = 64*t_{DTS}$
0100: $t_{DTS2} = 256*t_{DTS}$
0101: $t_{DTS2} = 1024*t_{DTS}$
0110: $t_{DTS2} = 4096*t_{DTS}$
0111: $t_{DTS2} = 16384*t_{DTS}$
1000: $t_{DTS2} = 65536*t_{DTS}$
1001: $t_{DTS2} = 262144*t_{DTS}$
Others: Reserved, do not program this value

Bits 23:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIM_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: $t_{DTS} = 8*t_{tim_ker_ck}$

Bit 7 **ARPE**: Auto-reload preload enable

0: TIM_ARR register is not buffered
1: TIM_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIM_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1).

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

45.8.2 TIM control register 2 (TIM_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

45.8.3 TIM DMA/interrupt enable register (TIM_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.
										rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bits 20:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

45.8.4 TIM status register (TIM_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			r				rc_w0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	BGF	SBIF	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	rc_w0	rc_w0				rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TI1FS**: tim_ti1f status

This flag indicates the value of the tim_ti1f signal (after the digital filtering stage), for polling purpose.

0: tim_ti1f signal is low

1: tim_ti1f signal is high

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **UIOVRF**: Update Interrupt Overrun flag

This flag is set by hardware when an update request occurs while an update interrupt is pending (update interrupt flag set). It is cleared by software by writing it to '0'.

0: No Update Interrupt Overrun

1: Update Interrupt Overrun

Bits 23:15 Reserved, must be kept at reset value.

Bit 14 **BGF**: Break generation flag

This flag is set by hardware as soon as a break event is forced by software, using the BG bit. It is useful to distinguish hardware-triggered from software-triggered break events. It is cleared by software by writing it to 0.

0: No break event occurred.

1: A break event was forced with the BG bit

Bit 13 **SBIF**: System break interrupt flag

This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.

This flag must be reset to re-start PWM operation.

0: No break event occurred.

1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIM_DIER register.

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIM_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIM_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIM_CNT matches the content of the TIM_CCR1 register. When the content of TIM_CCR1 is greater than the content of TIM_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIM_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIM_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIM_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIM_EGR register, if URS = 0 and UDIS = 0 in the TIM_CR1 register.

45.8.5 TIM event generation register (TIM_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIM_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

45.8.6 TIM capture/compare mode register 1 (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N =
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIM_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

45.8.7 TIM capture/compare mode register 1 [alternate] (TIM_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:18 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 17:16, 6:4 **OC1M[4:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

00000: Frozen - The comparison between the output compare register TIM_CCR1 and the counter TIM_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

00001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM_CNT matches the capture/compare register 1 (TIM_CCR1).

00011: Toggle - tim_oc1ref toggles when TIM_CNT = TIM_CCR1.

00100: Force inactive level - tim_oc1ref is forced low.

00101: Force active level - tim_oc1ref is forced high.

00110: PWM mode 1 - Channel 1 is active as long as TIM_CNT<TIM_CCR1 else inactive.

00111: PWM mode 2 - Channel 1 is inactive as long as TIM_CNT<TIM_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM_CCR1 disabled. TIM_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM_CCR1 enabled. Read/Write operations access the preload register. TIM_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIM_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM_CCER).

45.8.8 TIM capture/compare enable register (TIM_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 533* for details.

Table 533. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CC1E bit</th><th>CC1NE bit</th><th>tim_oc1 output state</th><th>tim_oc1n output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
tim_oc1 = 0
tim_oc1n = 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_oc1 = 0</td><td>tim_oc1ref + Polarity
tim_oc1n = tim_oc1ref XOR CC1NP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_oc1ref + Polarity
tim_oc1 = tim_oc1ref XOR CC1P</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_oc1n = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>tim_oc1ref + Polarity + dead-time</td><td>Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_oc1 = CC1P</td><td>tim_oc1ref + Polarity
tim_oc1n = tim_oc1ref XOR CC1NP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_oc1ref + Polarity
tim_oc1 = tim_oc1ref XOR CC1P</td><td>Off-State (output enabled with inactive state)
tim_oc1n = CC1NP</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z)</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: tim_oc1 = CC1P, tim_oc1n = CC1NP
Then if the clock is present: tim_oc1 = OIS1 and tim_oc1n = OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

45.8.9 TIM counter (TIM_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIM_ISR register. If the UIFREMAP bit in TIM_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

45.8.10 TIM prescaler (TIM_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM_EGR register or through trigger controller when configured in “reset mode”).

45.8.11 TIM autoreload register (TIM_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 45.4.3: Time-base unit on page 2294* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

45.8.12 TIM repetition counter register (TIM_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

45.8.13 TIM capture/compare register 1 (TIM_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

45.8.14 TIM break and dead-time register (TIM_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIM_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 45.8.8: TIM capture/compare enable register (TIM_CCER) on page 2414*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register) and when BKE bit is set.

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 45.8.8: TIM capture/compare enable register (TIM_CCER) on page 2414*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 45.8.8: TIM capture/compare enable register (TIM_CCER) on page 2414*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM_BDTR register, OISx and OISxN bits in TIM_CR2 register and BKBID/BKE/BKP/AOE bits in TIM_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$

Example if t_{DTS}=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

45.8.15 TIM timer deadtime register 2 (TIM_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{DTG} with $t_{DTG} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{DTG} with $t_{DTG} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{DTG} with $t_{DTG} = 16xt_{DTS}$.

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

45.8.16 TIM input selection register (TIM_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

45.8.17 TIM alternate function register 1 (TIM_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bit 31 **BKCMP20E**: tim_brk_cmp20 enable
This bit enables the tim_brk_cmp20 for the timer's tim_brk input. tim_brk_cmp20 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp20 input disabled
1: tim_brk_cmp20 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 30 **BKCMP19E**: tim_brk_cmp19 enable

This bit enables the tim_brk_cmp19 for the timer's tim_brk input. tim_brk_cmp19 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp19 input disabled

1: tim_brk_cmp19 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 29 **BKCMP18E**: tim_brk_cmp18 enable

This bit enables the tim_brk_cmp18 for the timer's tim_brk input. tim_brk_cmp18 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp18 input disabled

1: tim_brk_cmp18 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 28 **BKCMP17E**: tim_brk_cmp17 enable

This bit enables the tim_brk_cmp17 for the timer's tim_brk input. tim_brk_cmp17 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp17 input disabled

1: tim_brk_cmp17 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 27 **BKCMP16E**: tim_brk_cmp16 enable

This bit enables the tim_brk_cmp16 for the timer's tim_brk input. tim_brk_cmp16 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp16 input disabled

1: tim_brk_cmp16 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 26 **BKCMP15E**: tim_brk_cmp15 enable

This bit enables the tim_brk_cmp15 for the timer's tim_brk input. tim_brk_cmp15 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp15 input disabled

1: tim_brk_cmp15 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 25 **BKCMP14E**: tim_brk_cmp14 enable

This bit enables the tim_brk_cmp14 for the timer's tim_brk input. tim_brk_cmp14 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp14 input disabled

1: tim_brk_cmp14 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 24 **BKCMP13E**: tim_brk_cmp13 enable

This bit enables the tim_brk_cmp13 for the timer's tim_brk input. tim_brk_cmp13 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp13 input disabled

1: tim_brk_cmp13 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 23 **BKCMP12E**: tim_brk_cmp12 enable

This bit enables the tim_brk_cmp12 for the timer's tim_brk input. tim_brk_cmp12 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp12 input disabled

1: tim_brk_cmp12 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 22 **BKCMP11E**: tim_brk_cmp11 enable

This bit enables the tim_brk_cmp11 for the timer's tim_brk input. tim_brk_cmp11 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp11 input disabled

1: tim_brk_cmp11 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 21 **BKCMP10E**: tim_brk_cmp10 enable

This bit enables the tim_brk_cmp10 for the timer's tim_brk input. tim_brk_cmp10 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp10 input disabled

1: tim_brk_cmp10 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 20 **BKCMP9E**: tim_brk_cmp9 enable

This bit enables the tim_brk_cmp9 for the timer's tim_brk input. tim_brk_cmp9 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp9 input disabled

1: tim_brk_cmp9 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 19:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

45.8.18 TIM alternate function register 2 (TIM_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **OCRSEL[3:0]**: tim_ocref_clr source selection
These bits select the tim_ocref_clr input source.
0000: tim_ocref_clr0
0001: tim_ocref_clr1
...
1111: tim_ocref_clr15
Refer to *Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

45.8.19 TIM multilevel protection register 1 (TIM_MPR1)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK1NM[1:0]		BK1M[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **BK1NM[1:0]**: Break mode for tim_oc1n output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bits 1:0 **BK1M[1:0]**: Break mode for tim_oc1 output
This bit must be modified only when CEN=0 in TIM_CR1.
00: Immediate break
01: Delayed 1 break
10: Delayed 2 break
11: Reserved

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

45.8.20 TIM multilevel protection register 2 (TIM_MPR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DBK2[7:0]**: Delayed 2 Break duration

This bitfield defines the duration of the delay in Delayed 2 Break mode.
It is preloaded and can be updated during timer operation (new value taken into account after an update event).
DBK2[7:5]=0xx => Delay2=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay2=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay2=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DBK1[7:0]**: Delayed 1 Break duration

This bitfield defines the duration of the delay in Delayed 1 Break mode.
It is preloaded and can be updated during timer operation (new value taken into account after an update event).
DBK2[7:5]=0xx => Delay1=DBK2[7:0]xt_{DTG} with $t_{DTG}=t_{DTS}$.
DBK2[7:5]=10x => Delay1=(64+DBK2[5:0])xt_{DTG} with $t_{DTG}=2xt_{DTS}$.
DBK2[7:5]=110 => Delay1=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=8xt_{DTS}$.
DBK2[7:5]=111 => Delay1=(32+DBK2[4:0])xt_{DTG} with $t_{DTG}=16xt_{DTS}$.

Note: This bit can not be modified as long as LOCK level 2 or 3 has been programmed (LOCK bits in TIM_BDTR register).

45.8.21 TIM output override register (TIM_OOR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS1N	OOS1
			rw											rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **OOC**: Output override control

This bit forces the idle state of the tim_ocx and tim_ocxn outputs to the values defined with OOSx and OOSxN bits.

0: tim_ocx and tim_ocxn outputs are in the default idle state

1: tim_ocx and tim_ocxn outputs are forced in an idle state defined with OOSx/OOSxN bits

Note: This bit can only be modified when the outputs are in idle state (MOE = 0). It is cleared when the MOE bit is set.

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **OOS1N**: Output override state for output tim_oc1n

This bit defines the state for output tim_oc1n when OOC = 1

0: tim_oc1n = 0

1: tim_oc1n = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

Bit 0 **OOS1**: Output override state for output tim_oc1

This bit defines the state for output tim_oc1 when OOC = 1

0: tim_oc1 = 0

1: tim_oc1 = 1

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM_BDTR register).

45.8.22 TIM DMA control register (TIM_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM_DMAR address). DBA is defined as an offset starting from the address of the TIM_CR1 register.
Example:
00000: TIM_CR1,
00001: TIM_CR2,
00010: TIM_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIM_CR1. In this case the transfer is done to/from 7 registers starting from the TIM_CR1 address.

45.8.23 TIM DMA address for full transfer (TIM_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM_CR1 address) + (DBA + DMA index) x 4

where TIM_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM_DCR).

45.8.24 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 534. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM_CR1**	Res.	Res.	Res.	Res.	CKD2[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value					0	0	0	0												0	0		0	0	0	0	0	0	0	0	0	0
0x04	**TIM_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x08	Reserved	Res.																															
0x0C	**TIM_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value											0												0	0	0		0				0	0
0x10	**TIM_SR**	Res.	Res.	Res.	TI1FS	Res.	Res.	Res.	UIOVRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BGF	SBIF	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value				0				0										0	0				0		0		0				0	0
0x14	**TIM_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	**TIM_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIM_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[4:3]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value															0	0									0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	**TIM_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	**TIM_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 534. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**TIM_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIM_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	**TIM_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	**TIM_AF1**	BKCMP20E	BKCMP19E	BKCMP18E	BKCMP17E	BKCMP16E	BKCMP15E	BKCMP14E	BKCMP13E	BKCMP12E	BKCMP11E	BKCMP10E	BKCMP9E	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	**TIM_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0																
0x6C - 0x08C	Reserved	Res.																															
0x090	**TIM_MPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK1NM[1:0]		BK1M[1:0]	
	Reset value																													0	0	0	0
0x094	**TIM1x_MPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBK1[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x98 - 0x09C	Reserved	Res.																															

Table 534. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A0	**TIM_OOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OOS1N	OOS1
	Reset value																				0											0	0
0xA4 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

46 Low-power timer (LPTIM)

46.1 LPTIM introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

46.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by pulse counter application)
- 16-bit ARR autoreload register
- 16-bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable digital glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

46.3 LPTIM implementation

Table 535 describes LPTIM implementation on STM32G56x/55x/54x/53x devices.

Table 535. STM32G56x/55x/54x/53x LPTIM features

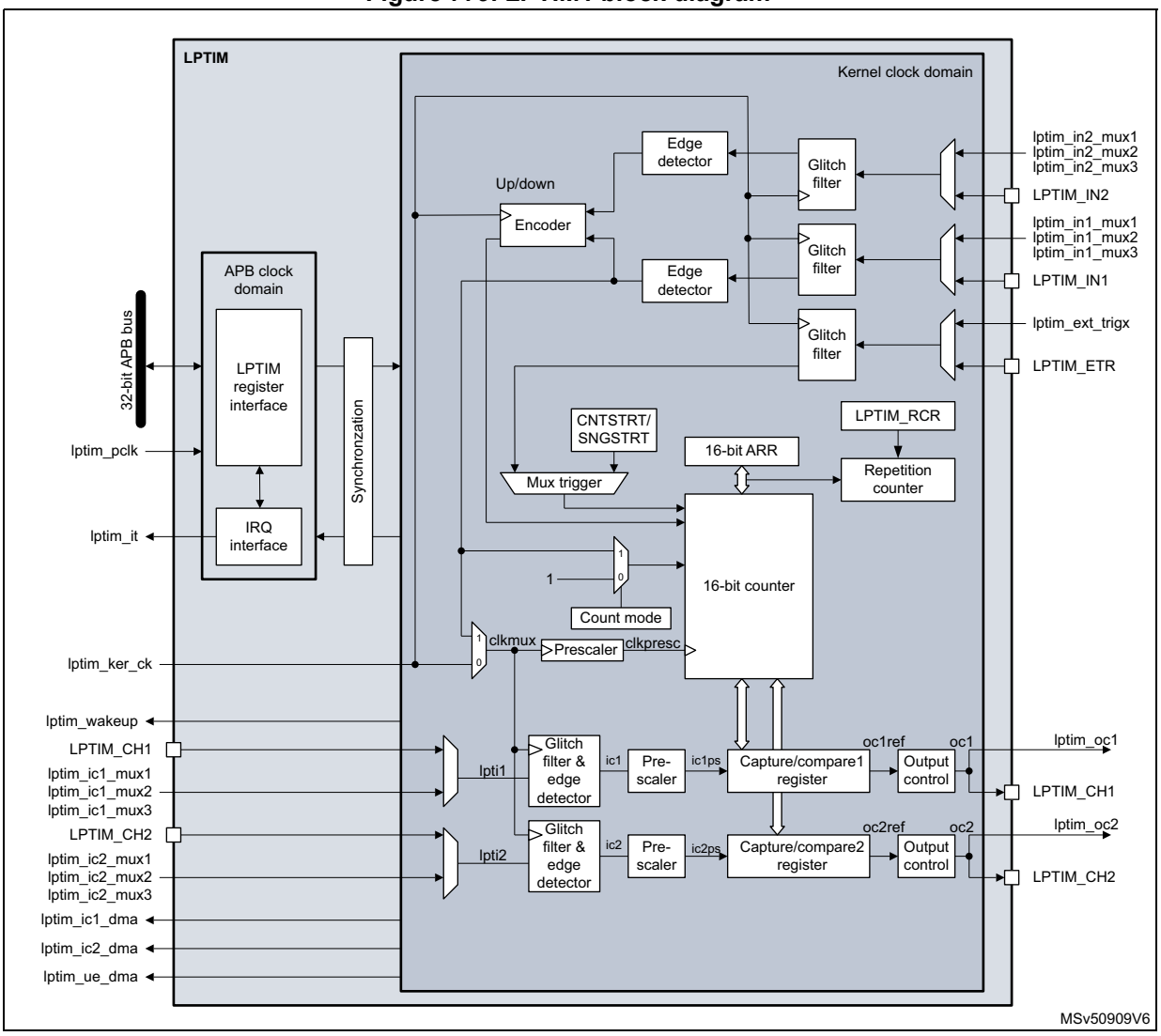
LPTIM modes/features(1)	LPTIM1
Encoder mode	X
PWM mode	X
Input Capture	X
Number of channels	2
Number of DMA requests	3
Wake-up from Stop mode	X
Autonomous mode	-

1. X = supported.

46.4 LPTIM functional description

46.4.1 LPTIM block diagram

Figure 779. LPTIM1 block diagram(1)



1. Some I/Os may not be available, refer to *Section 46.4.2: LPTIM pins and internal signals*.

46.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 536. LPTIM1 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0

Table 536. LPTIM1 input/output pins (continued)

Pin name	Pin type	Description
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIO pin

Table 537. LPTIM1 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request

46.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 538. LPTIM1 external trigger connections

TRIGSEL	External trigger
lptim_ext_trig0	LPTIM_ETR
lptim_ext_trig1	rtc_alra_trg

Table 538. LPTIM1 external trigger connections (continued)

TRIGSEL	External trigger
lptim_ext_trig2	rtc_alrb_trg
lptim_ext_trig3	tamp_trg1
lptim_ext_trig4	tamp_trg2
lptim_ext_trig5	gpdma1_ch1_tcf
lptim_ext_trig6	comp2_out
lptim_ext_trig7	play1_out10

Table 539. LPTIM1 input 1 connections

lptim_in1_mux	LPTIM1 input 1 connected to
lptim_in1_mux0	LPTIM_IN1
lptim_in1_mux1	comp1_out
lptim_in1_mux2	comp3_out
lptim_in1_mux3	comp5_out

Table 540. LPTIM1 input 2 connections

lptim_in2_mux	LPTIM1 input 2 connected to
lptim_in2_mux0	LPTIM_IN2
lptim_in2_mux1	comp2_out
lptim_in2_mux2	comp4_out
lptim_in2_mux3	comp6_out

Table 541. LPTIM1 input capture 1 connections

lptim_ic1_mux	LPTIM1 input capture 1 connected to
lptim_ic1_mux0	LPTIM_CH1
lptim_ic1_mux1	comp1_out
lptim_ic1_mux2	hsi_cal_ck
lptim_ic1_mux3	play1_out8

Table 542. LPTIM1 input capture 2 connections

lptim_ic2_mux	LPTIM1 input capture 2 connected to
lptim_ic2_mux0	LPTIM_CH2
lptim_ic2_mux1	LSI
lptim_ic2_mux2	LSE
lptim_ic2_mux3	play1_out9

46.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM can run in one of the following configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

46.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

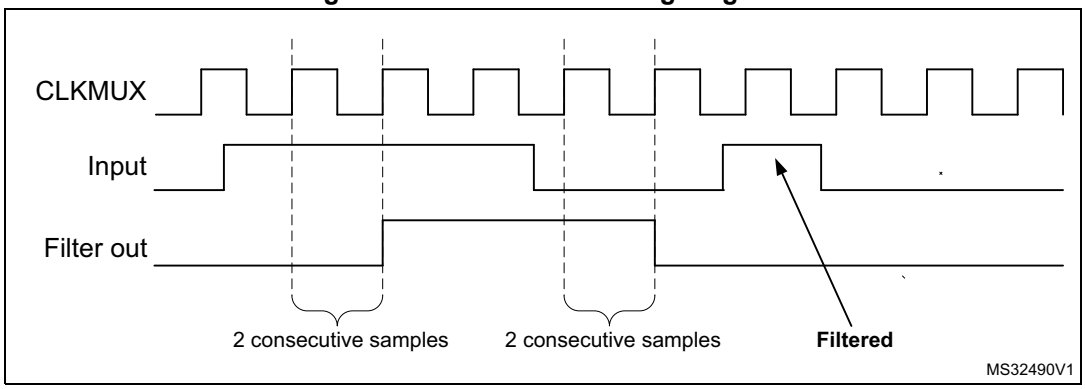
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 780* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 780. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to 0. In this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

46.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] field. The table below lists all the possible division ratios:

Table 543. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

46.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals 00, the LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than 00, TRIGSEL[2:0] is used to select which of the eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. After a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

46.4.8 Operating mode

The LPTIM features two operating modes:

- Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

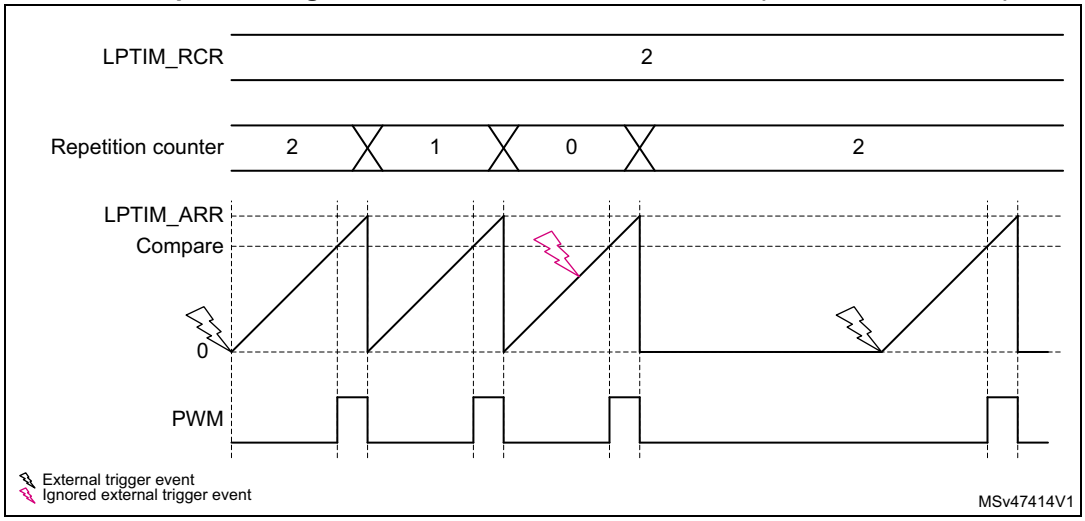
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 781*.

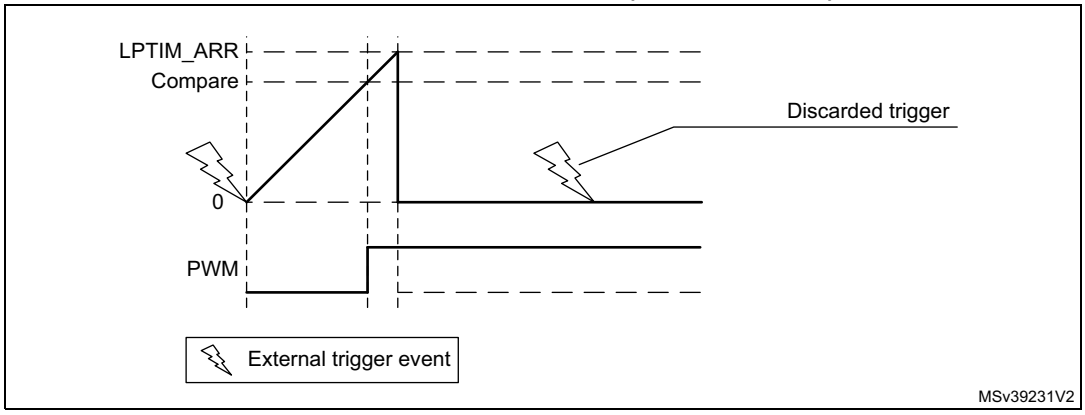
Figure 781. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 782*.

Figure 782. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = 00), the SNGSTRT setting starts the counter for one-shot counting.

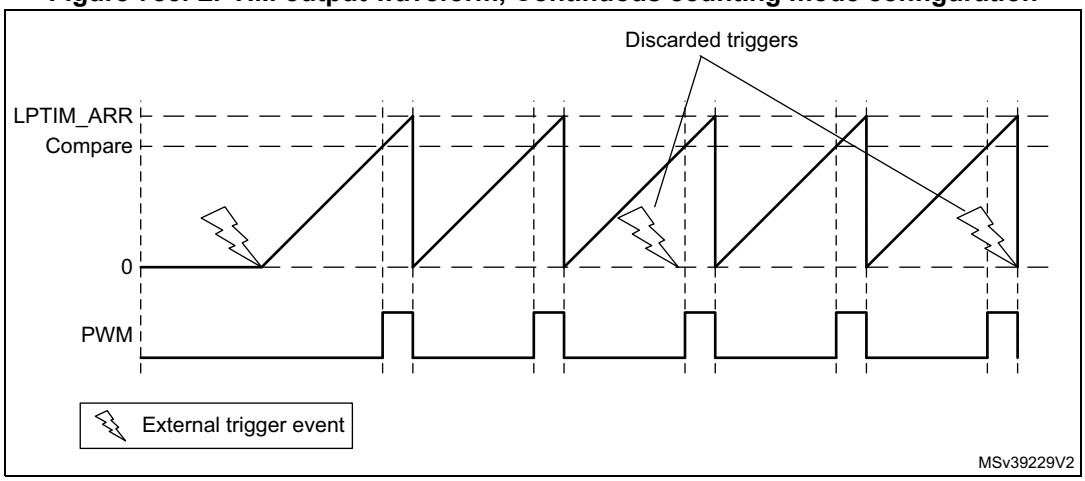
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 783*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 783. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit set to 1). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

46.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

46.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 46.4.19: PWM mode*.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

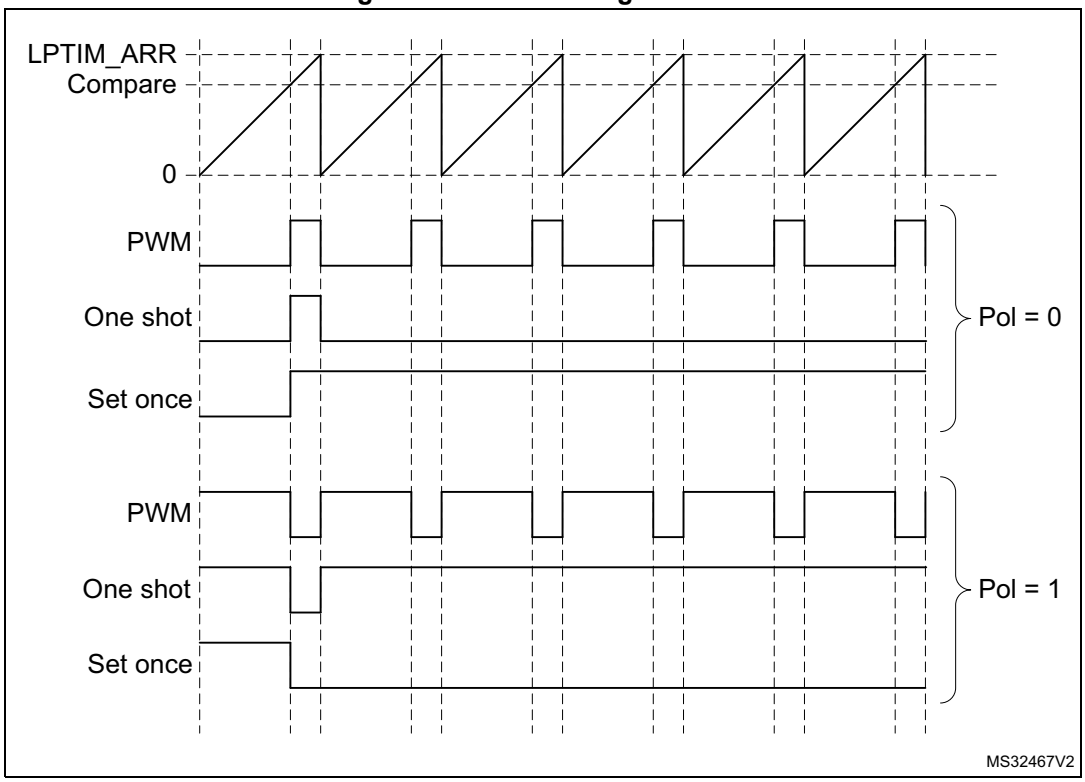
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to 0 forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to 1 forces the LPTIM to generate a Set-once mode waveform.

The CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by two can be generated. *Figure 784* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the CCxP bit.

Figure 784. Waveform generation



46.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to 0, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to 1, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

46.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

46.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

46.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is triggered and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

Note: The software ensures that COUNRST bit is '0' before generating every synchronous reset.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  To read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. The developer must make sure that these two mechanisms are used exclusively.

46.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two down and up flags in the LPTIM_ISR register. An interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to 1. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

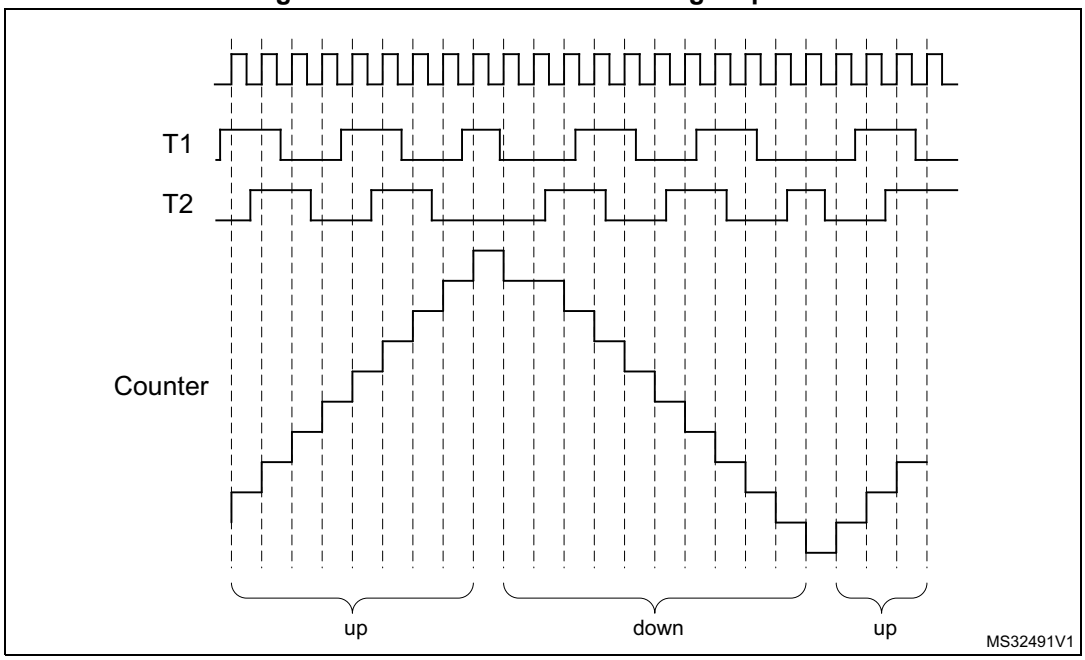
Table 544. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to 0. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be 000).

Figure 785. Encoder mode counting sequence



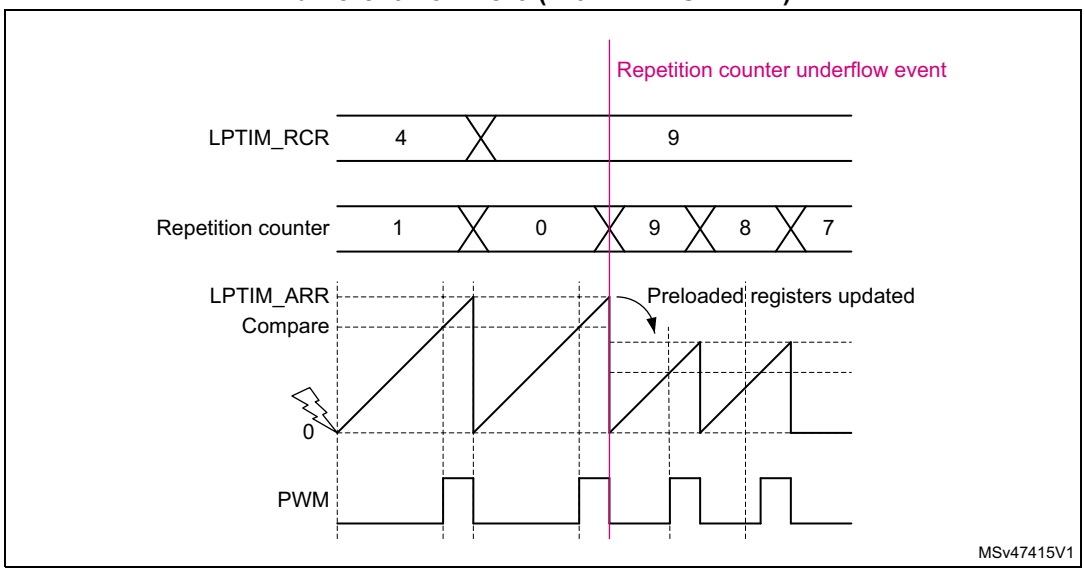
46.4.16 Repetition counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 786*.

Figure 786. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to *Section 46.4.11: Register update* for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive writes before the REPOK flag is set, lead to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

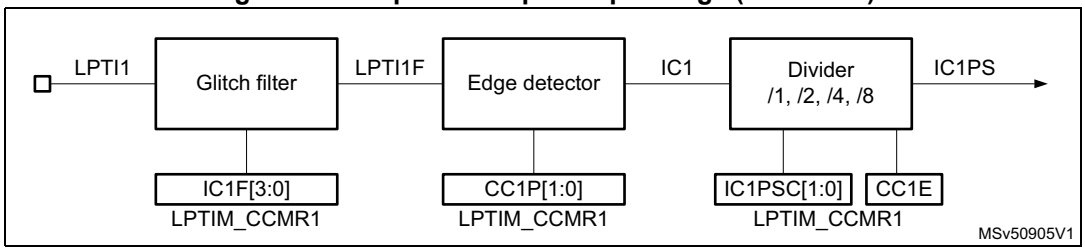
46.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

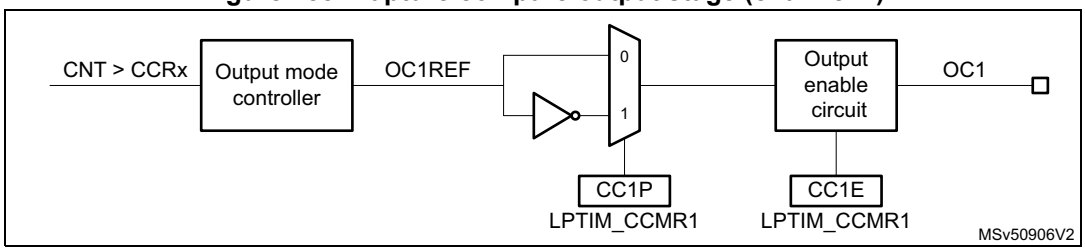
Figure 787. Capture/compare input stage (channel 1)



Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

Figure 788. Capture/compare output stage (channel 1)



46.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value (PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 545*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 545*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 545. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2
5	0	2
	1	2
	2	2
	3	2

Table 545. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

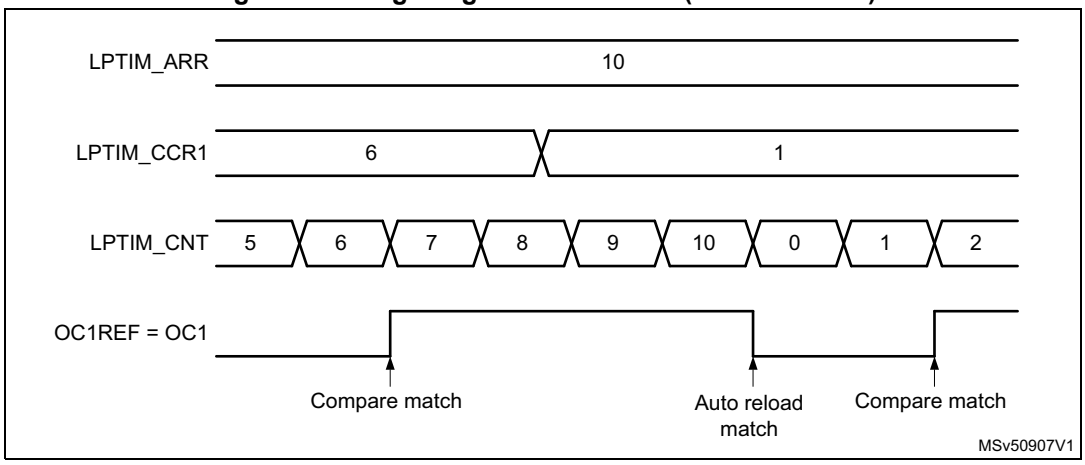
46.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 789 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

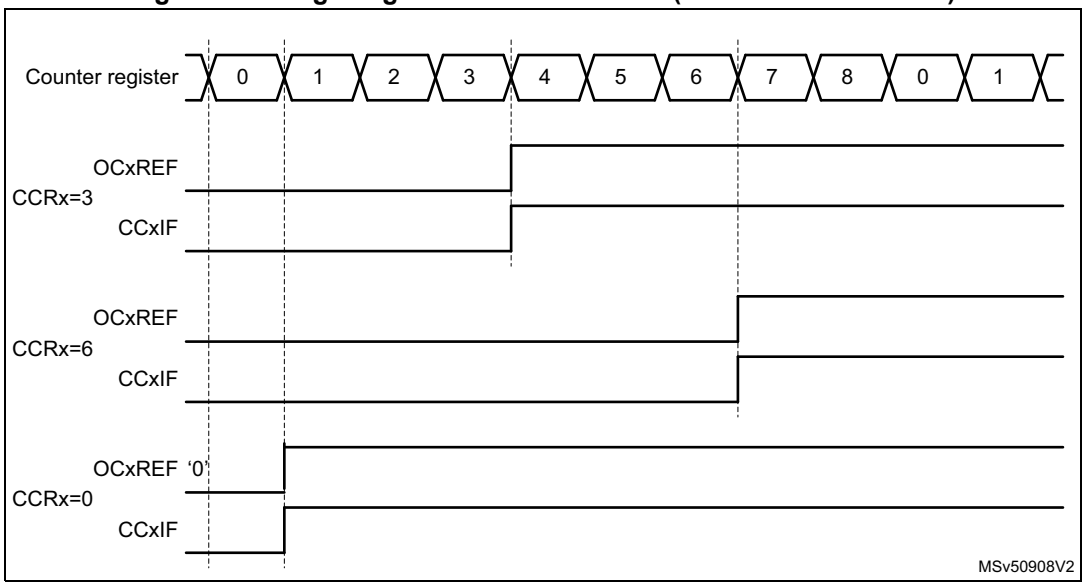
Figure 789. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 790 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 790. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

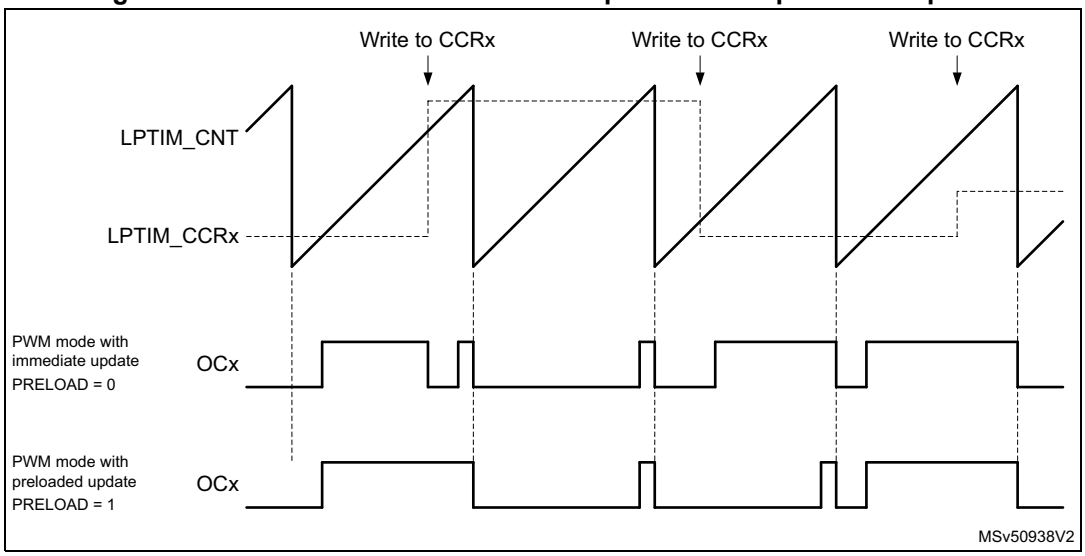
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 791* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 791. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match, and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 789 illustrates this behavior.

46.4.20 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, multiple times, at regular intervals, without software overhead

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the LPTIM_CCRx register. The captured values in LPTIM_CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in LPTIM_CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

46.4.21 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

46.5 LPTIM low-power modes

Table 546. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode.
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

Note: All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

46.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if Encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (Encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (status register) is set, the interrupt is not asserted.

Table 547. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(2)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(2)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to *Section 46.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 46.3: LPTIM implementation*.

46.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 111* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

46.7.1 LPTIM1 interrupt and status register [alternate] (LPTIM1_ISR)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	CMP2 OK	Res.	Res.	Res.
							r					r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

46.7.2 LPTIM1 interrupt and status register [alternate] (LPTIM1_ISR)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OF	CC1OF	Res.	Res.	CC2IF	REPOK	UE	DOWN	UP	ARROK	Res.	EXTTRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 12 **CC1OF**: Capture 1 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

46.7.3 LPTIM1 interrupt clear register [alternate] (LPTIM1_ICR)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	Res.
							w					w			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag

Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

46.7.4 LPTIM1 interrupt clear register [alternate] (LPTIM1_ICR)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2 OCF	CC1 OCF	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARRO KCF	Res.	EXTTR IGCF	ARRM CF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

46.7.5 LPTIM1 interrupt enable register [alternate] (LPTIM1_DIER)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE	Res.	Res.	Res.
								rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
0: CMPOK register 2 interrupt disabled
1: CMPOK register 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

46.7.6 LPTIM1 interrupt enable register [alternate] (LPTIM1_DIER)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

46.7.7 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	Res.	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw		rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: The counter is incremented following each internal clock pulse
1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 Reserved, must be kept at reset value.

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

0: A trigger event arriving when the timer is already started is ignored

1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

00: Software trigger (counting start is initiated by software)

01: Rising edge is the active edge

10: Falling edge is the active edge

11: Both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0

001: lptim_ext_trig1

010: lptim_ext_trig2

011: lptim_ext_trig3

100: lptim_ext_trig4

101: lptim_ext_trig5

110: lptim_ext_trig6

111: lptim_ext_trig7

See *Section 46.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1

001: /2

010: /4

011: /8

100: /16

101: /32

110: /64

111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any trigger active level change is considered as a valid trigger

01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.

10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.

11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any external clock signal level change is considered as a valid transition

01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: The rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 1 is active.

01: The falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 2 is active.

10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 3 is active.

11: Not allowed

Refer to *Section 46.4.15: Encoder mode* for more details about Encoder sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to 0).

46.7.8 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.

This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock can be different from APB clock).

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software must consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

46.7.9 LPTIM compare register 1 (LPTIM_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

46.7.10 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value

ARR is the autoreload value for the LPTIM.

This value must be strictly greater than the CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

46.7.11 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value

When the LPTIM is running, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

46.7.12 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 46.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 46.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection
The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 46.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection
The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 46.4.3: LPTIM input and trigger mapping*.

46.7.13 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

46.7.14 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		rw	rw			rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC2 rising edge

01: Falling edge, circuit is sensitive to IC2 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC1 rising edge

01: Falling edge, circuit is sensitive to IC1 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3) kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

46.7.15 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 46.3: LPTIM implementation*.

46.7.16 LPTIM register map

The following table summarizes the LPTIM registers.

Table 548. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM1_ISR Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIM1_ISR Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF(1)	CC1OF	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIM1_ICR Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIM1_ICR Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF(1)	CC1OCF	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIM1_DIER Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIM1_DIER Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE(1)	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE(1)	CC1OIE	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0

Table 548. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	Res.	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
	Reset value								0	0	0		0	0	0	0		0	0	0		0	0	0		0	0		0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	LPTIM_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C	LPTIM_CCMR1	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0
0x034	LPTIM_CCR2(3)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 46.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 46.3: LPTIM implementation*.
3. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 46.3: LPTIM implementation*.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

47 Independent watchdog (IWDG)

47.1 IWDG introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 47.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

47.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

47.3 IWDG implementation

Table 549. IWDG features (1)

IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	X
Capability to work in system Standby	X
Capability to generate an interrupt in system Stop(3)	X
Capability to generate an interrupt in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(4)	X
Option bytes to control the activity in Stop mode(5)	X
Option bytes to control the activity in Standby mode(6)	X
Option bytes to control the Hardware mode(7)	X

1. 'X' = supported, '-' = not supported.
2. Refer to the RCC section for additional information.
3. Wake-up from Stop with interrupt is supported only in Stop 0, Stop 1 mode.

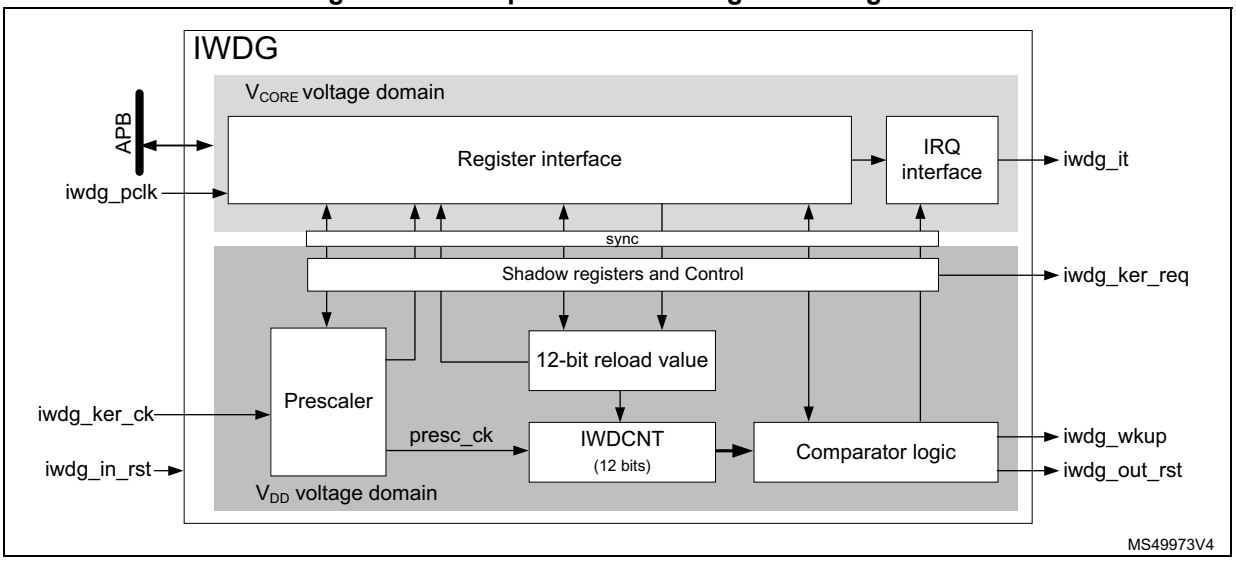
4. Controlled via DBG_IWDG_STOP in DBG section.
5. Controlled via the option byte IWDG_STOP in FLASH section.
6. Controlled via the option byte IWDG_STDBY in FLASH section.
7. Controlled via the option byte IWDG_SW in FLASH section.

47.4 IWDG functional description

47.4.1 IWDG block diagram

Figure 792 shows the functional blocks of the independent watchdog module.

Figure 792. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 47.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 550. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

47.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 551*.

Table 551. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

47.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is generally started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 47.3* for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

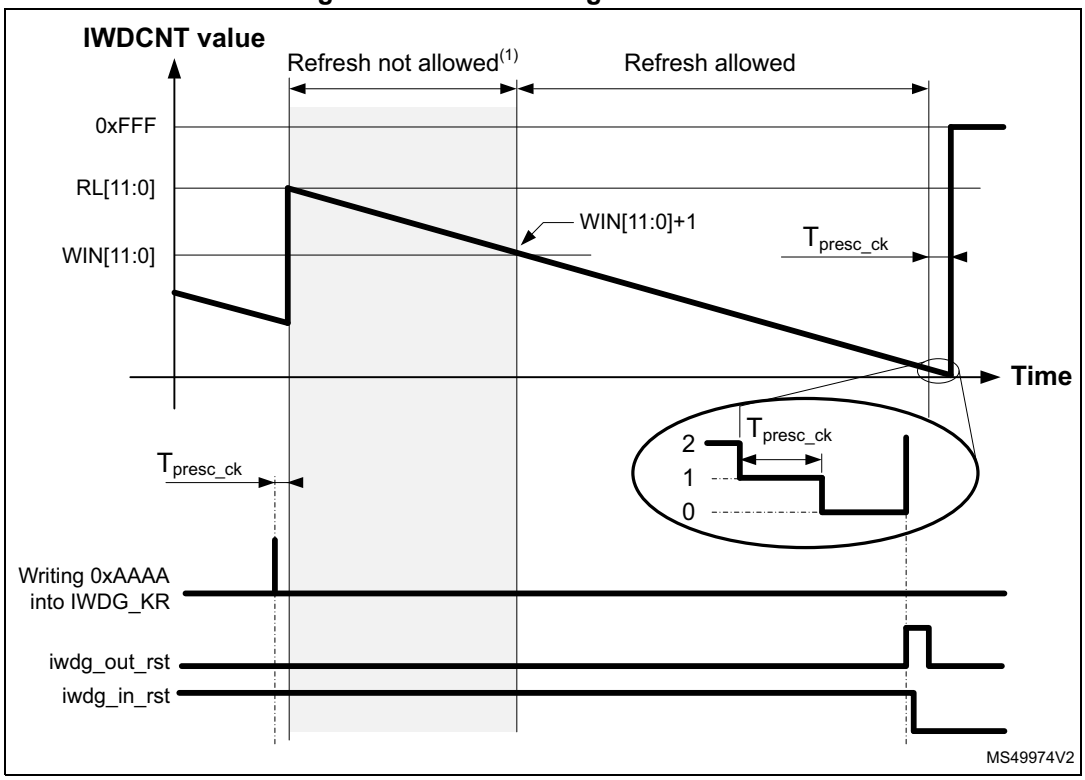
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 793*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 793. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

Note: The iwdg_out_rst signal typically triggers a system reset, however, this behavior may vary across products. For more details, refer to Chapter 47.3.

47.4.4 Window option

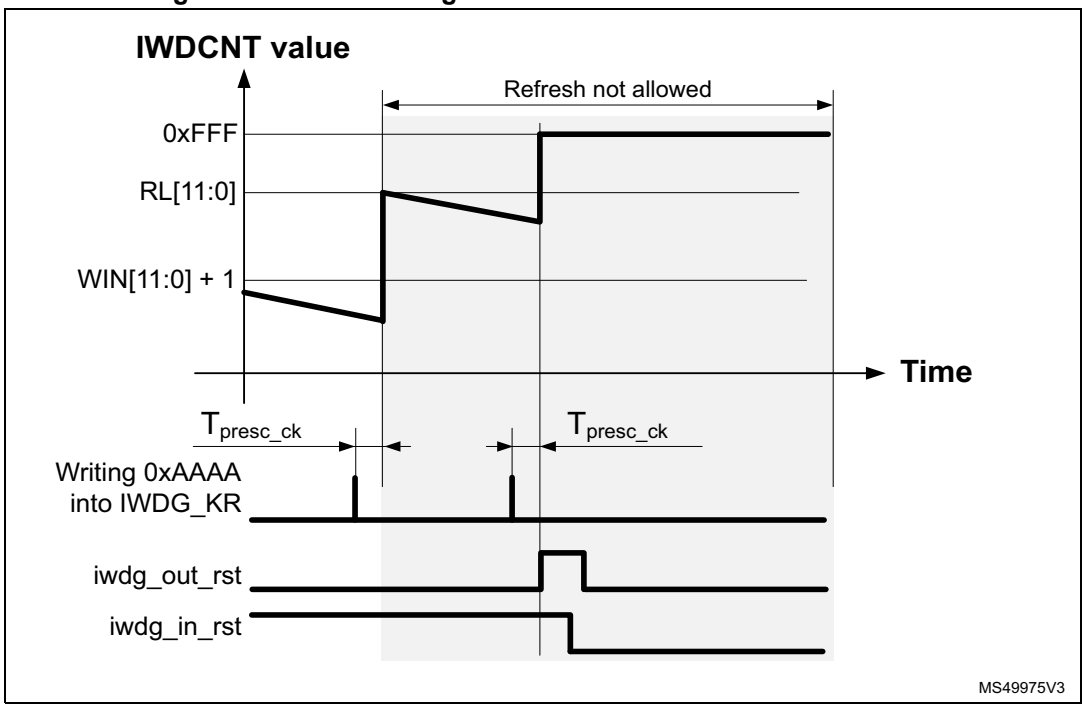
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 794*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 794. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0100).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

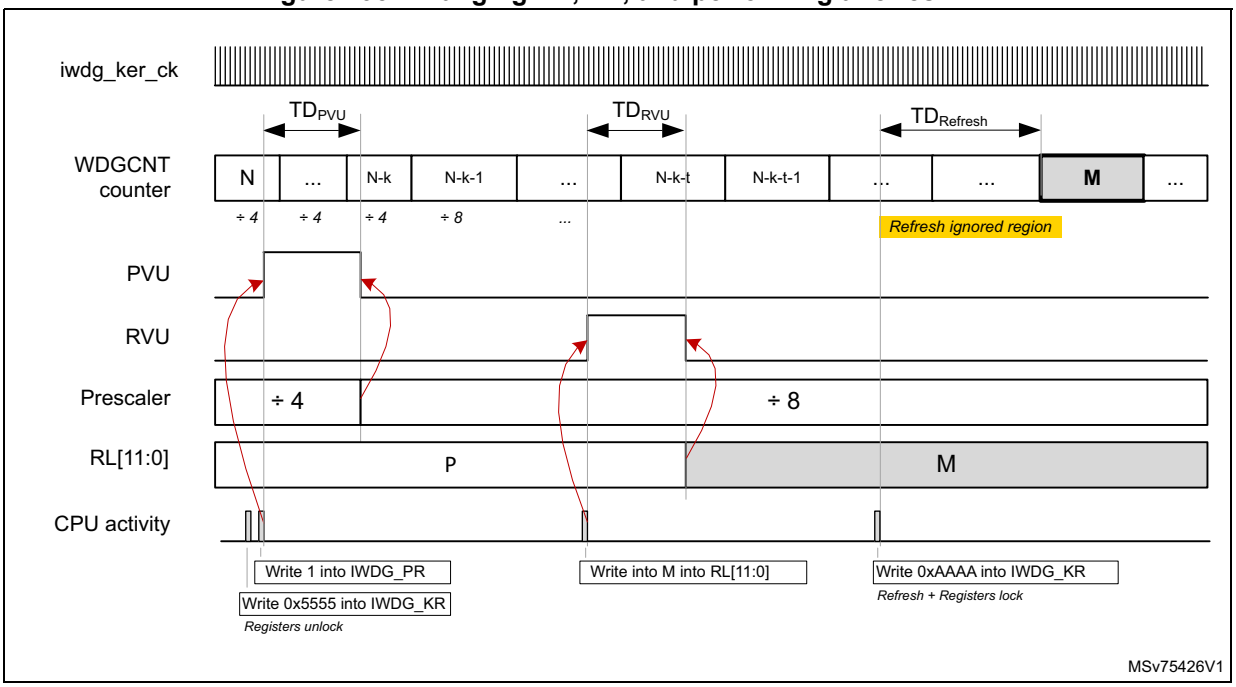
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0100).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 795. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 550: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

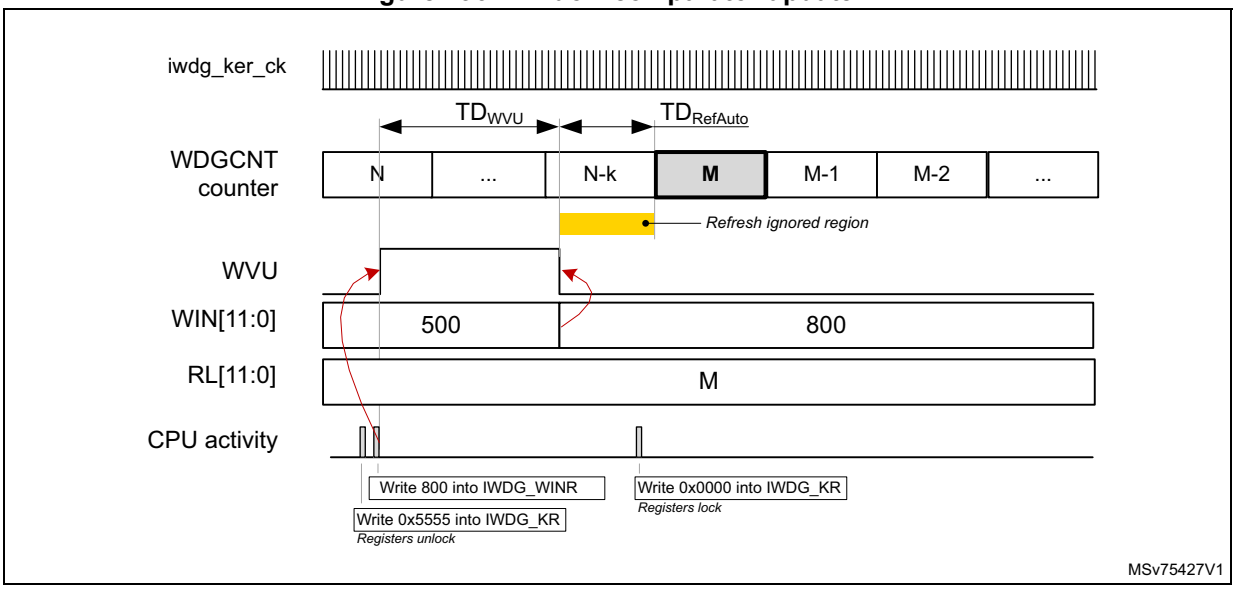
It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 796 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high for TDwvu. After a window comparator update, a refresh is automatically performed. The refresh is effective in the worst case, on the next prescaler clock active edge after the falling edge of WVU (TDRefAuto)

Figure 796. Window comparator update(1)



1. Refer to *Table 550: IWDG delays versus actions* for details on timing values.

47.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 47.3* for details on the capabilities of this product.

47.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the

sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

47.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 47.3* for details.

Table 552. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 47.3* for details. IWDG interrupts cause the device to exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 47.3* for details. IWDG interrupts do not make the device to exit from Standby mode.

47.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

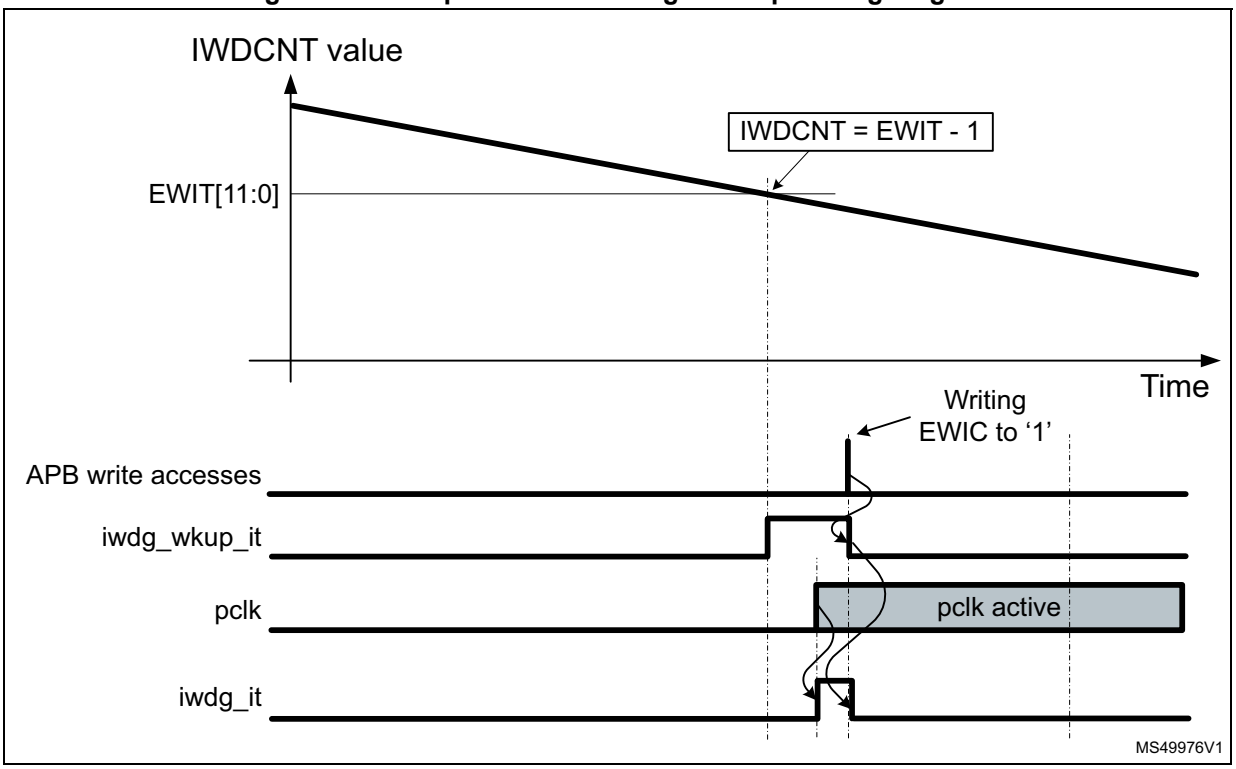
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG interrupt clear register (IWDG_ICR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 797. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

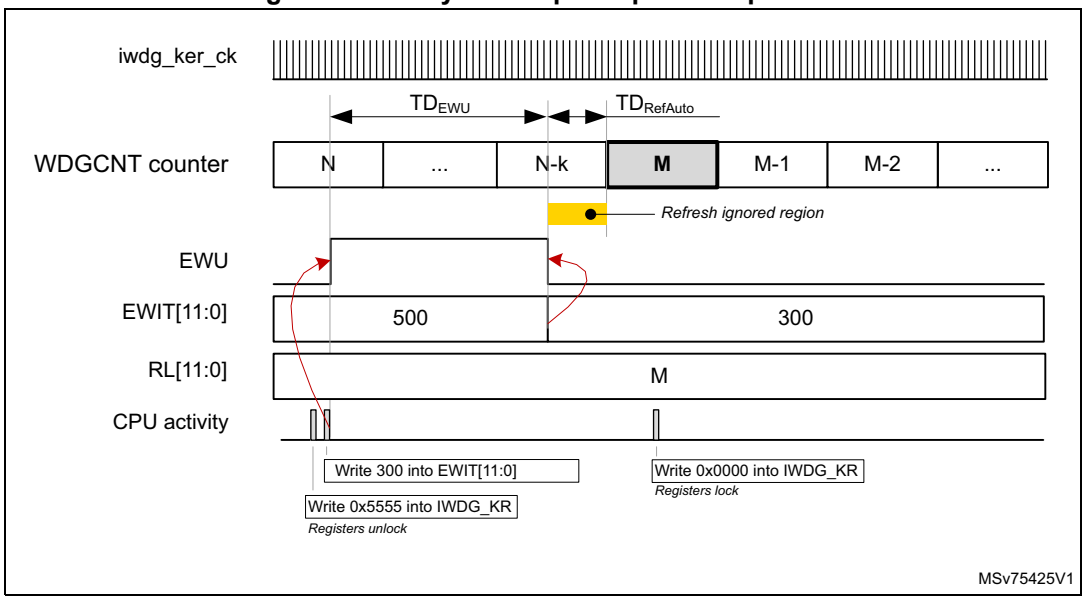
It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 798 shows this sequence. During the early wake-up comparator update operation the flag EWU remains to 1 for TDewu. A refresh is automatically performed. The refresh is effective on the worst case, on the next prescaler clock active edge after the falling edge of EWU (TDRefAuto).

Figure 798. Early wake-up comparator update(1)



1. Refer to *Table 550: IWDG delays versus actions* for details on timing values.

Table 553 summarizes the IWDG interrupt request.

Table 553. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

47.7 IWDG registers

Refer to *Section 1.2 on page 111* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

47.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

47.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 47.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

47.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 47.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

47.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in

software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
r							r					r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_ICR register to ‘1’.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 550: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 550: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 550: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 550: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the

early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

47.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 47.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

47.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIE	Res.	Res.	Res.	EWIT[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter early wake-up interrupt value

These bits are write access protected (see *Section 47.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

47.7.7 IWDG interrupt clear register (IWDG_ICR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 14:0 Reserved, must be kept at reset value.

47.7.8 IWDG register map

Table 554. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													0	0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																	0							x					0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	Res.	Res.	Res.	EWIT[11:0]											
	Reset value																	0				0	0	0	0	0	0	0	0	0	0	0	0
0x18	IWDG_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

48 System window watchdog (WWDG)

48.1 WWDG introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

48.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 800*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

48.3 WWDG implementation

Table 555. WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the hardware mode(4)	X

1. “X” = supported, “-” = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via WWDG_STOP in DBG block.

4. Controlled via the option byte WWDG_SW. When WWDG_SW is set in HW mode the WWDG is running as soon as the CPU is in Run or Sleep modes.

48.4 WWDG functional description

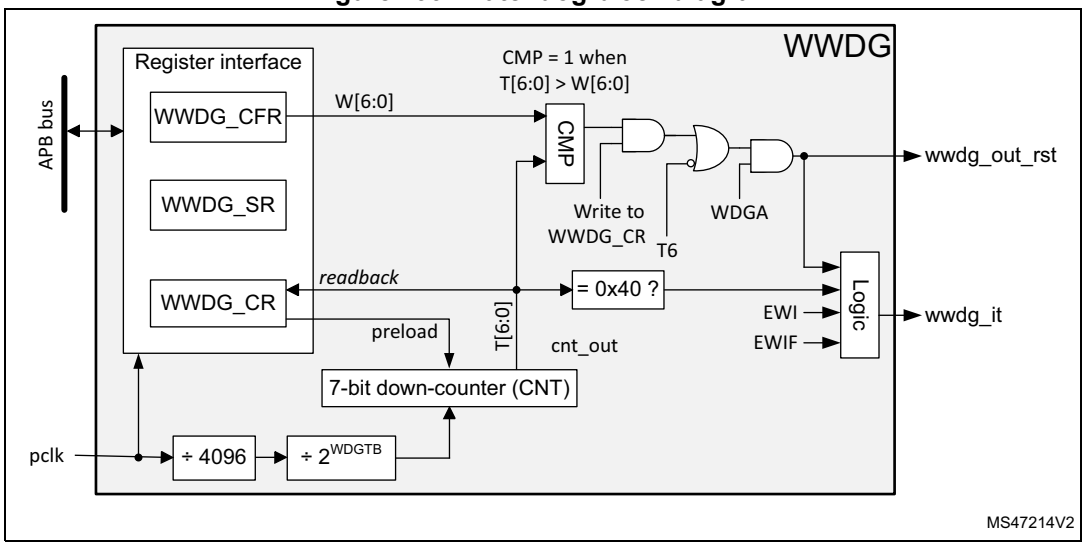
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 799* for the WWDG block diagram.

48.4.1 WWDG block diagram

Figure 799. Watchdog block diagram



48.4.2 WWDG internal signals

Table 556 gives the list of WWDG internal signals.

Table 556. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

48.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

Note: Refer to Section 48.3: WWDG implementation to check the availability of the hardware window watchdog mode.

48.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 800*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 800* describes the window watchdog process.

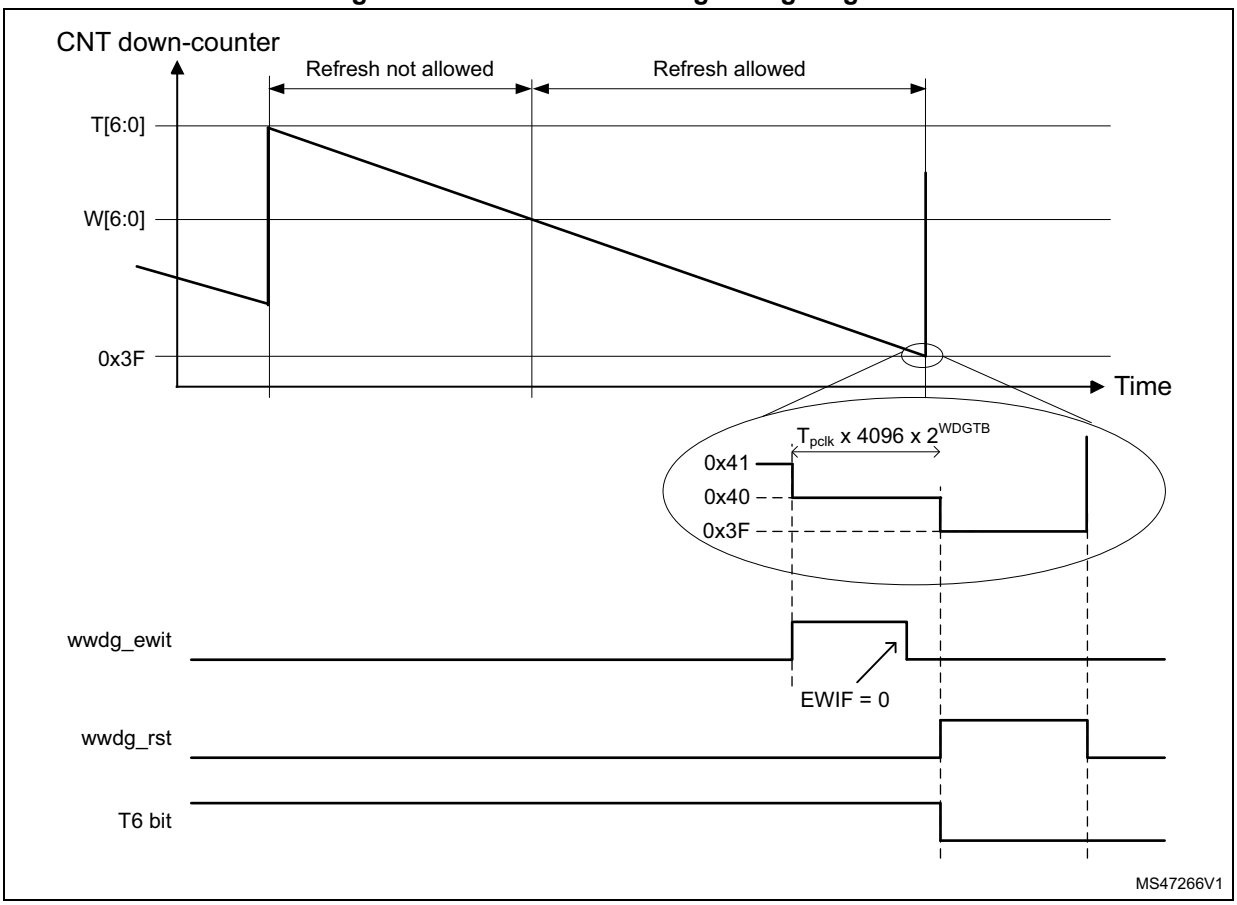
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

48.4.5 How to program the watchdog timeout

Use the formula in *Figure 800* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 800. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

48.4.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 60: Debug support (DBG)*.

48.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

Table 557. WWDG interrupt requests

Interrupt		Event flag	Enable control bit	Interrupt clearing method	Exit from mode		
Vector	Event				Sleep	Stop(1)	Standby(1)
WWDG(2)	Early wake-up interrupt	EWIF	EWI	Write EWIF flag to 0	Yes	No	No

1. The WWDG interrupt can have additional capabilities, refer to *Section 48.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

48.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

48.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

48.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

48.6.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

48.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 558. WWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

49 Real-time clock (RTC)

49.1 RTC introduction

The RTC provides an automatic wake-up feature to manage low-power modes.

The real-time clock (RTC) is an independent counter that supports the BCD format. It provides a time-of-day clock and calendar with programmable alarm interrupts and timestamp.

49.2 RTC main features

The RTC supports the following features (see *Figure 801: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wake-up Timer and timestamp individual secure or nonsecure configuration
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

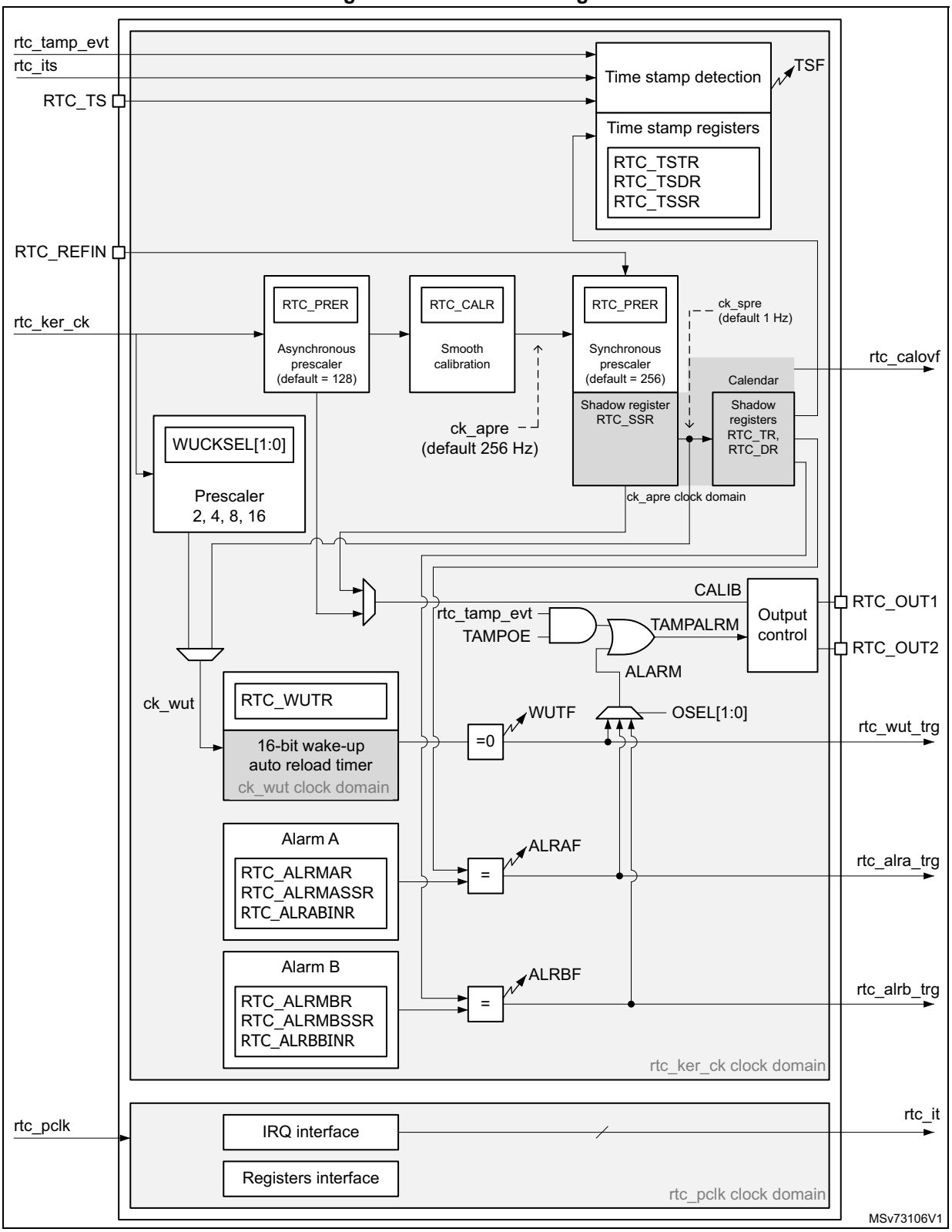
The RTC is functional in the low-power modes described in the *Section 49.4: RTC low-power modes*. It is not reset by a system reset NRST.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the supported low-power modes.

49.3 RTC functional description

49.3.1 RTC block diagram

Figure 801. RTC block diagram



49.3.2 RTC pins and internal signals

Table 559. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which selects one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 560. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_it	Output	RTC interrupts (refer to *Section 49.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes when the selected clock is not LSE. Refer to *Section 49.4: RTC low-power modes* for more details.

The RTC is reset by software by setting the RTCDRST bit in RCC.

Table 561. RTC interconnection

Signal name	Source/destination
rtc_its	Unused
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

49.3.3 GPIOs controlled by the RTC

The GPIOs with RTC functions available in Standby mode are directly controlled by the RTC, whatever the GPIO configuration.

The different RTC functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, TAMPOE, COE and OUT2EN configuration, as shown in *Table 562*.

Table 562. RTC_OUT mapping

OSEL[1:0] bits	TAMPOE	COE bit	OUT2EN bit	RTC_OUT1	RTC_OUT2
00	0	0	0	-	-
		1		CALIB	-
Don't care	1	Don't care		TAMPALRM	-
01 or 10 or 11	Don't care				-
00	0	0	1	-	-
		1		-	CALIB
Don't care	1	0		-	TAMPALRM
01 or 10 or 11	Don't care			-	
Don't care	1	1		TAMPALRM	CALIB
01 or 10 or 11	Don't care				

49.3.4 RTC registers write protection

After a system reset, the RTC registers are protected against parasitic write access by the DRTCP bit in the power control peripheral (refer to the PWR power control section). DRTCP bit must be set to enable RTC registers write access.

49.3.5 RTC secure protection modes

By default after a RTC domain reset, all RTC registers can be read or written in both secure and nonsecure modes, except for the RTC secure configuration register (RTC_SECCFGR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the SEC bit is set in the RTC_SECCFGR register:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When the SEC bit is cleared, it is still possible to protect some of the registers by setting dedicated INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC or ALRBSEC control bits. If all these bits are also clear, all the RTC registers can be read and written in secure and nonsecure mode.

- When INITSEC is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When CALSEC is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When ALRASEC is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_SMISR can be read and written only in secure mode.
  - ALRAPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When ALRBSEC is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_SMISR can be read and written only in secure mode.
  - ALRBPRIV in the RTC_PRIVCFGR can be written only in secure mode.
- When WUTSEC is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_SMISR can be read and written only in secure mode.
  - WUTPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When TSSEC is set:
  - RTC_TSTR, RTC_TSDR, RTC_TSSSR, RTC_TAMPTSCR and RTC_TSIDR registers, plus TAMPTS, TSE, TSIE, TSEDGE control bits in the RTC_CR, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF in RTC_SR, and TSMF, TSOVMF in RTC_SMISR can be read and written only in secure mode.
  - TSPRIV in the RTC_PRIVCFGR can be written only in secure mode

A nonsecure access to a secure-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

49.3.6 RTC privilege protection modes

By default after a RTC domain reset, all RTC registers can be read or written in both privileged and unprivileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and unprivilege modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and unprivilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALDSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRASEC in the RTC_SECCFGR can be written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in

RTC_SR, and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.

  - ALRBSEC in the RTC_SECCFGR can be written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - WUTSEC in the RTC_SECCFGR can be written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR, RTC_TSSSR, RTC_TAMPTSCR and RTC_TSIDR registers, plus TAMPTS, TSE, TSIE, TSEDGE control bits in the RTC_CR, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF in RTC_SR, and TSMF, TSOVMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - TSSEC in the RTC_SECCFGR can be written only in privilege mode.

An unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

49.3.7 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into two programmable prescalers (see *Figure 801: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 49.3.11: Periodic auto-wake-up* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

49.3.8 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 49.6.12: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the

BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least seven times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

49.3.9 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL = 0 to LPCAL = 1 or from LPCAL = 1 to LPCAL = 0 is not immediate and requires a few ck_apre periods to complete.

49.3.10 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least three to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL of the RTC_CR register.

49.3.11 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in two parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 2516*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 49.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

49.3.12 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC write protection key

After RTC domain reset, the RTC registers related to RTC initialization and RTC calibration are write-protected: Writing the unlock sequence in RTC_WPR is required to gain write access to these registers.

The following steps are required to unlock the write protection on the registers related to RTC initialization that is RTC_TR, RTC_DR, RTC_PRER, the bits INIT, BIN and BCDU in RTC_ICSR, and the bit FMT in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be performed with the privilege and security mode defined by PRIV, INITPRIV, SEC, INITSEC configuration

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with incorrect privilege, security settings has no effect and is ignored without any illegal access notification.

The following steps are required to unlock the write protection of the registers related to RTC calibration, that is RTC_CALR, RTC_SHIFTR, and the bits SUB1H, ADD1H, REFCKON in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be done with the privilege and security mode defined by PRIV, CALPRIV, SEC, CALSEC configuration.

Writing a wrong key reactivates the write protection.

Any write access to RTC_WPR with the wrong privilege, security settings has no effect and is ignored without any illegal access notification.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   – If LPCAL = 0: INITF is set around two RTCCLK cycles after INIT bit is set.
   – If LPCAL = 1: INITF is set up to two ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   – If LPCAL = 0: the counting restarts after four RTCCLK clock cycles.
   – If LPCAL = 1: the counting restarts after up to two RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN = 00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN = 01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: There is no hardware protection against incorrect calendar value initialization. Software must ensure that the initialization is performed correctly using allowed calendar BCD values.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its RTC domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around two RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed.
   - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

49.3.13 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than two times the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the three values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than one RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 2514*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 49.3.15: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the two readings. If the values differ, repeat the process until matching results are obtained. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

49.3.14 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a RTC domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the RTC domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a RTC domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

49.3.15 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of one ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN = 10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

49.3.16 RTC reference clock detection

This feature is available only in BCD mode (BIN = 00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock is higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals seven ck_apre periods when detecting the first reference clock edge. A smaller window of three ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

49.3.17 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL = 0: ck_cal = RTCCLK

If LPCAL = 1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL = 0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) is reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S is set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S is set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to two RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16-second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the

calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during eight seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Recalibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

49.3.18 Timestamp function

Timestamp is enabled by setting the TSE bit of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS or any bit of RTC_TAMPTSCR is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to *RTC control register (RTC_CR)* and *RTC timestamp on tamper control register (RTC_TAMPTSCR)*.

When a timestamp event occurs, the timestamp flag bit (TSF) in RTC_SR register is set. The TSID bitfield in the RTC_TSIDR provides the identification of the cause of the timestamp event. It is mandatory to check ITSF after clearing TSF status flag, in case an internal event occurs after first timestamp event. If set, ITSF must also be cleared by software to allow next timestamp event correct identification.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set up to two ck_apre cycles after the timestamp event from RTC_TS pin occurs, due to synchronization process. TSF is set up to three ck_apre cycles after tamper flags. TSID indicates the source corresponding to time stored in timestamp registers, but due to this difference of timing and priority between events, RTC_TS or rtc_its can be identified as first event, even if it occurs up to one ck_apre after TAMP event.

TSOVF is set up to only one ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

49.3.19 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

49.3.20 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

49.4 RTC low-power modes

Table 563. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 564. RTC pins functionality over modes

Functions	Functional in all low-power modes except Standby mode	Functional in Standby mode
RTC_TS	Yes	Yes
RTC_REFIN	Yes	No
RTC_OUT1	Yes	Yes
RTC_OUT2	Yes	No

49.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The nonsecure interrupt output or the secure interrupt output is also activated.

The RTC provides an interrupt to a specific CPU only if its CID filter is disabled, or if it is enabled and configured with the CPU CID in the RTC_RxCIDCFGR register (x = 0 to 3).

Table 565. Nonsecure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC	Alarm A	ALRAF	ALRAIE and (ALRASEC = 0 and SEC = 0)	Write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC = 0 and SEC = 0)	Write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC = 0 and SEC = 0)	Write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC = 0 and SEC = 0)	Write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC = 0 and SEC = 0)	Write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

Table 566. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC_S	Alarm A	ALRAF	ALRAIE and (ALRASEC = 1 or SEC = 1)	Write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=1 or SEC = 1)	Write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC = 1 or SEC = 1)	Write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC = 1 or SEC = 1)	Write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC = 1 or SEC = 1)	Write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_SMISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

49.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

49.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 2514* and *Reading the calendar on page 2516*.

This register is write protected. The write access procedure is described in *RTC write protection key on page 2514*.

This register can be write-protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x00

RTC domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

49.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 2514* and *Reading the calendar on page 2516*.

This register is write protected. The write access procedure is described in *RTC write protection key on page 2514*.

This register can be write-protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x04

RTC domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

49.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

RTC domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values

When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):

SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN = 00):

SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Subsecond value/synchronous binary counter LSB values

When Binary mode is selected (BIN = 01 or 10 or 11):

SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN = 00):

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

49.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC write protection key on page 2514*.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x0C

RTC domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Recalibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.
0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).
01: Free running Binary mode (BCD mode disabled)
10: Free running BCD calendar and Binary modes
11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Note: RSF is not set when the frequency of the APB clock is less than twice the frequency of RTCCLK.

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (RTC domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wake-up timer configuration update not allowed

1: Wake-up timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

49.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 2514*.

This register is write protected. The write access procedure is described in *RTC write protection key on page 2514*.

This register can be write-protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x10

RTC domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

49.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x14

RTC domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUTOCLR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

49.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC write protection key on page 2514.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x18

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after one ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.

1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after one ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM

1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved if none of the RTC_TAMPTSCR bit is set

1: Save RTC timestamp on tamper detection event (for any tamper event, whatever RTC_TAMPTSCR value)

TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to three ck_apre cycles after the tamper flags.

Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **COE**: Calibration output enable

This bit enables the CALIB output

0: Calibration output disabled

1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to TAMPALRM output.

00: Output disabled

01: Alarm A output enabled

10: Alarm B output enabled

11: Wake-up output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of TAMPALRM output.

0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection

When COE = 1, this bit selects which signal is output on CALIB.

0: Calibration output is 512 Hz

1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 49.3.19: Calibration clock output*.

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, one hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time.

Note: This can be used for summer time change
ADD1H is ignored when SUB1H is set.

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp interrupt disable

1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable

0: Wake-up timer interrupt disabled

1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable

0: Alarm B interrupt disable

1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: Timestamp disable

1: Timestamp enable

Bit 10 **WUTE**: Wake-up timer enable

0: Wake-up timer disabled

1: Wake-up timer enabled

Note: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable

0: Alarm B disabled

1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable

0: Alarm A disabled

1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable

0: SSR underflow interrupt disabled

1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC_REFIN detection disabled

1: RTC_REFIN detection enabled

Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge

0: RTC_TS input rising edge generates a timestamp event

1: RTC_TS input falling edge generates a timestamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.

11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

49.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged. This register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the RTC_SECCFGR configuration (refer to *Section 49.3.6: RTC privilege protection modes*).

Address offset: 0x1C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV	INIT PRIV	CAL PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS PRIV	WUT PRIV	ALRB PRIV	ALRA PRIV
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection

0: All RTC registers can be written when the APB access is privileged or unprivileged, except the registers protected by other privilege protection bits.

1: All RTC registers can be written only when the APB access is privileged.

Bit 14 **INITPRIV**: Initialization privilege protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or unprivileged.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, daylight saving, calibration and reference clock privilege protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is privileged or unprivileged.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection

0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wake-up timer privilege protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection

0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A and SSR underflow privilege protection

0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 49.3.6: RTC privilege protection modes for details on the read protection.

49.6.9 RTC secure configuration register (RTC_SECCFGR)

This register can be written only when the APB access is secure.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 49.3.6: RTC privilege protection modes*).

Address offset: 0x20

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC	INIT SEC	CAL SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS SEC	WUT SEC	ALRB SEC	ALRA SEC
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SEC**: RTC global protection

0: All RTC registers can be written when the APB access is secure or nonsecure, except the registers protected by other secure protection bits.

1: All RTC registers can be written only when the APB access is secure.

Bit 14 **INITSEC**: Initialization protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or nonsecure.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

Bit 13 **CALSEC**: Shift register, daylight saving, calibration and reference clock protection
0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or nonsecure.
1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSSEC**: Timestamp protection
0: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

Bit 2 **WUTSEC**: Wake-up timer protection
0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is secure.

Bit 1 **ALRBSEC**: Alarm B protection
0: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

Bit 0 **ALRASEC**: Alarm A and SSR underflow protection
0: RTC alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is secure.

Note: Refer to Section 49.3.5: RTC secure protection modes for details on the read protection.

49.6.10 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key
This byte is written by software.
Reading this byte always returns 0x00.
Refer to *RTC write protection key* for a description of how to unlock RTC register write protection.

49.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC write protection key on page 2514*.

This register can be write-protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x28

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).
This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.
Refer to *Section 49.3.17: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period
When CALW8 is set to 1, the 8-second calibration cycle period is selected.
Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 49.3.17: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period
When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.
Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 49.3.17: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature must be used in conjunction with CALP. See *Section 49.3.17: RTC smooth digital calibration on page 2519*.

49.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC write protection key on page 2514*.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = 1 - (SUBFS / (PREDIV_S + 1)).

In mixed BCD-binary mode (BIN = 10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

49.6.13 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

49.6.14 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

49.6.15 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x38

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

49.6.16 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x40

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don't care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don't care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don't care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

49.6.17 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x44

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN = 00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

49.6.18 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x48

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don’t care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don’t care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don’t care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

49.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x4C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].
Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN = 00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.

49.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x50

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	Res.	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag

This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.

Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload. SSRUF is set only in binary and mixed modes. It is not set in BCD mode.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.

If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.

If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

49.6.21 RTC nonsecure masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x54

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSR UMF	Res.	TSOV MF	TS MF	WUT MF	ALRB MF	ALRA MF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow nonsecure masked flag
This flag is set by hardware when the SSR underflow nonsecure interrupt occurs.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVMF**: Timestamp overflow nonsecure masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp nonsecure masked flag
This flag is set by hardware when a timestamp nonsecure interrupt occurs.

Bit 2 **WUTMF**: Wake-up timer nonsecure masked flag
This flag is set by hardware when the wake-up timer nonsecure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B nonsecure masked flag
This flag is set by hardware when the alarm B nonsecure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A nonsecure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

49.6.22 RTC secure masked interrupt status register (RTC_SMISR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x58

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow secure masked flag
This flag is set by hardware when the SSR underflow secure interrupt occurs.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVMF**: Timestamp overflow interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs.

Bit 2 **WUTMF**: Wake-up timer interrupt secure masked flag
This flag is set by hardware when the wake-up timer secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

49.6.23 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x5C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	Res.	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w		w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing '1' in this bit clears the SSRUF in the RTC_SR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSF bit in the RTC_SR register.

Bit 2 **CWUTF**: Clear wake-up timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

49.6.24 RTC timestamp on tamper control register (RTC_TAMPTSCR)

Address offset: 0x64

RTC domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMPTS
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2TS	TAMP1TS
														rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ITAMPTS**: Timestamp on internal tamper event
0: When TAMPTS = 0, internal tamper detection event does not cause a RTC timestamp to be saved
1: Save RTC timestamp on any internal tamper detection event (whatever TAMPTS value)

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **TAMP2TS**: Timestamp on external tamper TAMP2 event
0: When TAMPTS = 0, TAMP2 detection event does not cause an RTC timestamp to be saved
1: Save RTC timestamp on TAMP2 detection event (whatever TAMPTS value)

Bit 0 **TAMP1TS**: Timestamp on external tamper TAMP1 event
0: When TAMPTS = 0, TAMP1 detection event does not cause an RTC timestamp to be saved
1: Save RTC timestamp on TAMP1 detection event (whatever TAMPTS value)

Note: All bits of RTC_TAMPTSCR must be cleared before entering RTC initialization mode.

49.6.25 RTC timestamp status register (RTC_TSIDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR.

Address offset: 0x68

RTC domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSID[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TSID[5:0]**: Timestamp flag source identification
This bitfield is updated with identification of the source generating the timestamp stored in timestamp registers. It is updated at each TSF event, and is not affected by TSOVF event.
000000: RTC_TS pin event
000001: TAMP1F event
000010: TAMP2F event
010000: Internal tamper event
Others: Reserved

49.6.26 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x70

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.
SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSR, and so can also be read or written through RTC_ALRMASSR.

49.6.27 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 49.3.5: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 49.3.6: RTC privilege protection modes*.

Address offset: 0x74

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSR, and so can also be read or written through RTC_ALRMBSSR.

49.6.28 RTC register map

Table 567. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	Res.	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x20	RTC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	INITSEC	CALSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSSEC	WUTSEC	ALRBSEC	ALRASEC
	Reset value																	0	0	0										0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 567. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	RTC_TSSSR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]			MSK1	ST[2:0]			SU[3:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]			MSK1	ST[2:0]			SU[3:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	Res.	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0		0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0		0	0	0	0	0
0x58	RTC_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0		0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	Res.	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0		0	0	0	0	0

Table 567. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x64	RTC_ TAMPTSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMPTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2TS	TAMP1TS
	Reset value																0															0	0
0x68	RTC_TSIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSID[5:0]					
	Reset value																											0	0	0	0	0	0
0x70	RTC_ ALRABINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	RTC_ ALRBBINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

50 Tamper and backup registers (TAMP)

50.1 TAMP introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. Up to 32x 32-bit backup registers are retained in all low-power modes. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with four tamper pins (two input and two output pins) and up to 14 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

50.2 TAMP main features

- A tamper detection can optionally erase the backup registers, SRAM1, SRAM2, cryptographic peripherals. The device resources protected by tamper are named “device secrets”. The list of device secrets is configurable by software.

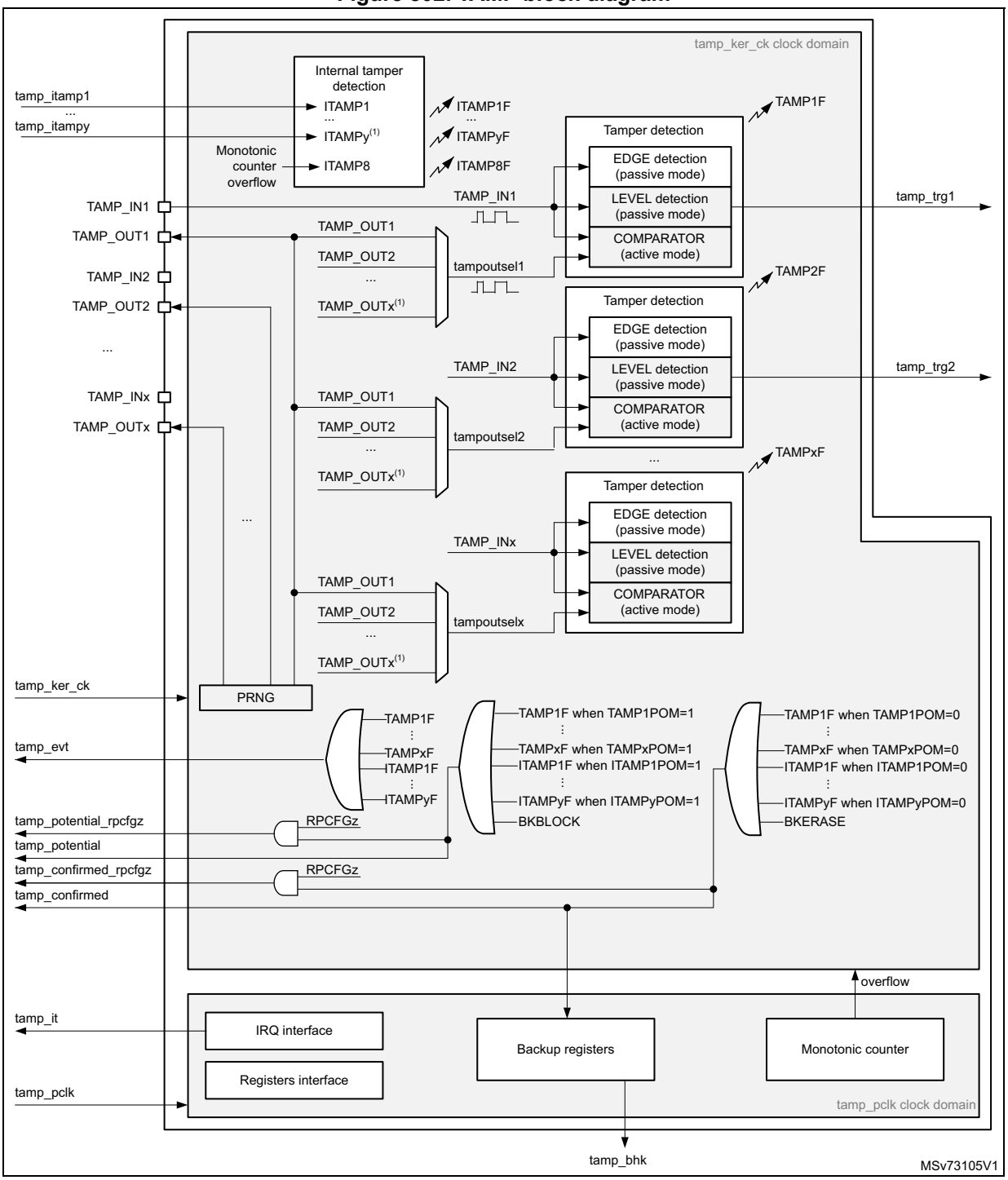
Up to 32×32-bit backup registers (16 for STM32G53x/54x devices, 32 for STM32G55x/56x devices)

- Up to four tamper pins (two input and two output pins) for two external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- Up to 14 internal tamper events to protect against transient or environmental perturbation attacks (12 events for STM32G53x/54x devices, 14 events for STM32G55x/56x devices)
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- TrustZone support:
  - Tamper secure or nonsecure configuration
  - Backup registers configuration in three configurable-size areas:
    1 read/write secure area
    1 write secure/read nonsecure area
    1 read/write nonsecure area
  - Boot hardware key for secure AES, stored in backup registers, protected against read and write access.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.

50.3 TAMP functional description

50.3.1 TAMP block diagram

Figure 802. TAMP block diagram



1. The number of external and internal tampers depends on products.

50.3.2 TAMP pins and internal signals

Table 568. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin
TAMP_OUTx (x = pin index)	Output	Tamper output pin (active mode only)

Table 569. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_tzen	Input	TAMP TrustZone enabled
tamp_evt	Output	Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets protection. This signal is active when: a tamper event detection flag (internal or external tamper), is generated in potential mode. or a software request is done by writing BKBLOCK to 1
tamp_confirmed	Output	Confirmed tamper detection signal, used for device secrets protection. This signal is active when: a tamper event detection flag (internal or external tamper), is generated in confirmed mode. or a software request is done by writing BKERASE to 1
tamp_potential_rpcfgz (z = signal index)	Output	Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: a tamper event detection flag (internal or external tamper), is generated in potential mode. or a software request is done by writing BKBLOCK to 1
tamp_confirmed_rpcfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: a tamper event detection flag (internal or external tamper), is generated in confirmed mode. or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 50.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger
tamp_bhk	Output	Tamper boot hardware key bus

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). The TAMP kernel clock is required only for External tamper inputs level with filtering, and for external active tamper detection modes. Internal tampers detection and external tampers inputs edge detection are functional without requiring any kernel clock. Read and write access to backup registers and all other TAMP registers are also functional without any kernel clock (only APB clock is needed).

Some detections modes are not available in some low-power modes depending on the selected clock (refer to *Section 50.4* for more details).

Table 570. TAMP interconnection

Signal name	Source/Destination
tamp_tzen	From flash option bytes
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_potential	No effect.
tamp_confirmed	No effect.
tamp_potential_rpcfg0	No effect.
tamp_confirmed_rpcfg0	No effect.
tamp_potential_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg1 signal is used to block the read and write access to the device secrets listed hereafter: SRAM2
tamp_confirmed_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg1 signal is used to erase the device secrets listed hereafter: SRAM2 The device secrets access is blocked when erase is ongoing.
tamp_potential_rpcfg2	When the bit RPCFG2 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg2 signal is used to block the device secrets listed hereafter: RHUK and to erase the device secrets listed hereafter: AES, SAES(1), CCB(1)
tamp_confirmed_rpcfg2	When the bit RPCFG2 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg2 signal is used to block the device secrets listed hereafter: RHUK and to erase the device secrets listed hereafter: AES, SAES(1), CCB(1)
tamp_potential_rpcfg3	When the bit RPCFG3 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg3 signal is used to erase the device secrets listed hereafter: ICACHE The device secrets access is blocked when erase is ongoing.
tamp_confirmed_rpcfg3	When the bit RPCFG3 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg3 signal is used to erase the device secrets listed hereafter: ICACHE The device secrets access is blocked when erase is ongoing.

Table 570. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_potential_rpcfg4	When the bit RPCFG4 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg4 signal is used to erase the device secrets listed hereafter: HASH
tamp_confirmed_rpcfg4	When the bit RPCFG4 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg4 signal is used to erase the device secrets listed hereafter HASH
tamp_potential_rpcfg5	When the bit RPCFG5 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg5 signal is used to erase the device secrets listed hereafter PKA including PKA SRAM The device secrets access is blocked when erase is ongoing.
tamp_confirmed_rpcfg5	When the bit RPCFG5 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg5 signal is used to erase the device secrets listed hereafter PKA including PKA SRAM The device secrets access is blocked when erase is ongoing.
tamp_potential_rpcfg6	When the bit RPCFG6 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg6 signal is used to erase the device secrets listed hereafter: DCACHE(1) The device secrets access is blocked when erase is ongoing.
tamp_confirmed_rpcfg6	When the bit RPCFG6 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg6 signal is used to erase the device secrets listed hereafter: DCACHE(1) The device secrets access is blocked when erase is ongoing.
tamp_potential_rpcfg7	No effect
tamp_confirmed_rpcfg7	When the bit RPCFG7 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg7 signal is used to erase the device secrets listed hereafter: OTFDEC KEYS(1) and CRC registers
tamp_potential_rpcfg8	When the bit RPCFG8 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg8 signal is used to block the read and write access to the device secrets listed hereafter: SRAM1
tamp_confirmed_rpcfg8	When the bit RPCFG8 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg8 signal is used to erase the device secrets listed hereafter: SRAM1 The device secrets access is blocked when erase is ongoing.
tamp_itamp1	V_{DD} domain monitoring(2)
tamp_itamp2	Temperature monitoring(2)
tamp_itamp3	LSE monitoring (LSECSS)(3)
tamp_itamp4	HSE monitoring (rcc_hsecss_fail)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	Unexpected debug activation
tamp_itamp7	ADC2 analog watchdog 1
tamp_itamp8(4)	Monotonic counter 1 overflow

Table 570. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_itamp9	Cryptographic peripherals fault (AES, SAES(1) or PKA or TRNG)
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)
tamp_itamp12	ADC2 analog watchdog 2
tamp_itamp13	ADC3(1) analog watchdog 1
tamp_itamp14	ADC3(1) analog watchdog 2
tamp_itamp15	System fault (PAR)
tamp_bhk	saes_bhk. This bus is used to load the boot hardware key in the secure AES co-processor.

1. Not present in STM32G53x/54x devices.
2. This monitoring must be enabled by setting MONEN in PWR tamper monitoring control register 1 (PWR_TMCR).
3. This monitoring must be enabled by setting LSECSSON in RCC RTC domain control register (RCC_RTCCR).
4. This signal is generated in the TAMP peripheral.

50.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the RTC domain are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_TS, TAMP_IN2, and TAMP_OUT1 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes.

The output mechanism follows the priority order shown in the table below.

Table 571. RTC pin PC13 configuration(1)

PC13 pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E = TAMP1AM = 1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE = 1 and ATOSELx = 1	TAMP2E (TAMP_IN2 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull		01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT1 output PP		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
TAMP_IN2 input floating		00	0	0	Don't care	Don't care	Don't care	0	1	0
		00	0	1	1					
		Don't care	Don't care	0						

Table 571. RTC pin PC13 configuration(1) (continued)

PC13 pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E = TAMP1AM = 1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE = 1 and ATOSELx = 1	TAMP2E (TAMP_IN2 input enable)	TSE (RTC_TS input enable)
RTC_TS and TAMP_IN2 input floating	00	0	0	Don't care	Don't care	Don't care	0	1	1
	00	0	1	1			0		
	Don't care	Don't care	0				0		
RTC_TS input floating	00	0	0	Don't care	Don't care	Don't care	0	0	1
	00	0	1	1			0		
	Don't care	Don't care	0				0		
Wake-up pin or standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0	0
	00	0	1	1			0		
	Don't care	Don't care	0				0		

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

50.3.4 TAMP register write protection

After system reset, TAMP registers (including backup registers) are protected against parasitic write access by the DRTCP bit in the power control peripheral (refer to the PWR section). DRTCP bit must be set in order to enable TAMP registers write access.

50.3.5 TAMP secure protection modes

Unless otherwise specified, after a backup domain reset, all TAMP registers are readable and writable in both secure and nonsecure modes. For information about specific bitfields or

complete registers that are writable only in secure mode, see *Section 50.6: TAMP registers*. The TAMP protection configuration is not affected by a system reset.

- When the TAMPSEC bit is set in TAMP_SECCFGR:
  - Writing the TAMP registers is possible only in secure mode, except for the backup registers and monotonic counter which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR, and TAMP_MISR is always possible in secure and nonsecure modes. All other TAMP registers can be read only in secure mode, except backup registers and monotonic counters, which have their own protection setting.
- When the CNT1SEC bit is set in TAMP_SECCFGR, TAMP_COUNT1R can be read and written only in secure mode, and CNT1PRIV can be written only in secure mode.
- When the BKPRWSEC is set in the TAMP_SECCFGR register, the BKPRWPRIV can be written only in secure mode.
- When the BKPWSEC is set in the TAMP_SECCFGR register: the BKPRWPRIV and BKPWPRIV can be written only in secure mode.

A nonsecure access to a secure-protected register is denied:

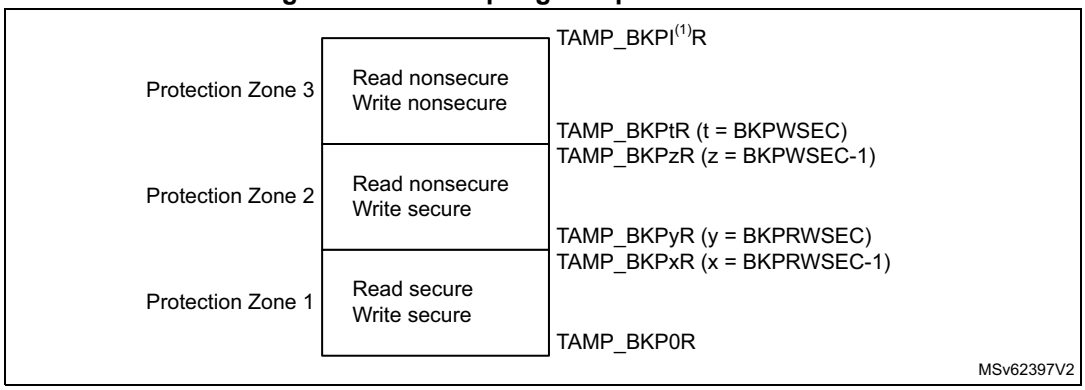
- There is no bus error generated.
- When write protected, bits are not written.
- When read protected, they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

50.3.6 Backup register protection zones

The backup register protection is configured by using BKPRWSEC[7:0] and BKPWSEC[7:0] (refer to the figure below).

Figure 803. Backup register protection zones



50.3.7 TAMP privilege protection modes

Unless otherwise specified, after a backup domain reset, all TAMP registers are readable and writable in both secure and nonsecure modes. For information about specific bitfields or

complete registers that are writable only in privilege mode, see *Section 50.6: TAMP registers*. The TAMP protection configuration is not affected by a system reset.

- When the TAMPPRIV bit is set in TAMP_PRIVCFGR:
  - Writing TAMP registers is possible only in privilege mode, except for backup registers and monotonic counters, which have their own protection settings.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR, is always possible in privileged and unprivileged modes. All the other TAMP registers can be read only in privileged mode, except for backup registers and monotonic counters which have their own protection setting.
- When the CNT1PRIV bit is set in TAMP_PRIVCFGR: the TAMP_COUNT1R can be read and written only in privilege mode, and CNT1SEC can be written only in privilege mode.
- When the BKPRWPRIV is set in TAMP_PRIVCFGR: the BKPRWSEC can be written only in privilege mode.
- When the BKPWPRIV is set in TAMP_PRIVCFGR: the BKPRWSEC and BKPWSEC can be written only in privilege mode.

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWSEC[7:0], BKPWSEC[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 803*). The BHKLOCK bit can be written only in privileged mode when the BKPRWPRIV bit is set.

An unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

50.3.8 Boot hardware key (BHK)

The eight first backup registers from TAMP_BKP0R to TAMP_BKP7R can be used to store a boot hardware key for the secure AES.

For this purpose, these registers must belong to the protection zone 1: BKPRWSEC must be greater or equal to 8.

Once backup registers are written with the boot hardware key, the BHKLOCK bit must be set in TAMP_SECCFGR. Once BHKLOCK is set, the eight backup registers cannot be accessed anymore by software: they are read as 0 and write to these registers is ignored. BHKLOCK cannot be cleared by software, and is cleared by hardware following a confirmed tamper event (if RPCFG31 = 1). It is also cleared with BKERASE command if RPCFG31 = 1 (in all cases, backup registers are also erased).

Refer to *Section 35: Secure AES coprocessor (SAES)* for details on procedure to download the boot hardware key in the SAES.

50.3.9 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- Erase backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 570*. The list of device secrets is configurable in TAMP_RPCFGR.
- Block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 570*. The list of device secrets is configurable thanks to TAMP_RPCFGR.
- Generate an interrupt that is able to wake up from low-power modes.
- Generate a hardware trigger for the low-power timers, or a RTC timestamp event.

The external I/O tamper detection supports two main configurations:

- Passive mode: TAMP_INx I/Os are monitored, and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x = pin index) is continuously compared with TAMP_OUTy (y = pin index) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection.
In addition, it is possible to configure each tamper source in potential mode, so that the secret erase is not launched by hardware on tamper detection. The secret erase can then be launched by software after software checks.

50.3.10 TAMP backup registers and other device secret erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

Backup registers and other device secrets are not reset when the corresponding mask is set (TAMPxMSK = 1 in TAMP_CR2).

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in TAMP_CR2. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in TAMP_CR3. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 570*.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in TAMP_CR2. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in TAMP_CR3. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in *Table 570*.

This mode is selected to avoid irreversible erase of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected.

In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in TAMP_SCR. Therefore software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in TAMP_CR2.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 570*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device resource protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode).
- It is not affected by BKERASE software command.
- It is not affected by BKBLOCK software command.

When RPCFGz = 1 in TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in *Table 570*.
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 570*.

Table 572. Backup registers and BHK tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFG31 = 0	No effect on backup registers, BHK, and BHKLOCK	No effect on backup registers, BHK, and BHKLOCK
RPCFG31 = 1	Backup registers and BHK access are blocked	Backup registers, BHK, and BHKLOCK are erased.

Table 573. Other device resource x tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFGx = 0	No effect on device resource x	No effect on device resource x
RPCFGx = 1	Device secret x protected as described by tamp_potential_rpcfgx(1)	Device secret x protected as described by tamp_confirmed_rpcfgx(1)

1. Refer to *Table 570*.

Device secret access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secret access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit in TAMP_CR2. The device secret access is possible only when BKBLOCK = 0 and no tamper flag is set.

50.3.11 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in TAMP_CR.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in TAMP_SR.

By setting the TAMPxIE bit in TAMP_IER, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR rester, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, backup registers are not cleared.

This feature is available only when the tamper is configured in *Level detection with filtering on tamper inputs (passive mode)* (TAMPFLT ≠ 00 and active mode not selected).

Timestamp on tamper event

With TAMPTS set to 1 in RTC_CR, any internal or external tamper event causes a timestamp to occur. When TAMPTS is 0, any external tamper can be individually configured to cause or not a timestamp. A global configuration for all internal tampers is also available to enable or not timestamp on internal tamper detection (refer to RTC_TAMPTSCR). If a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, as if a normal timestamp event occurs. The TSID bitfield in RTC_TSIDR provides the identification of the cause of the timestamp event.

Caution: Timestamp on tamper event must be disabled before entering RTC initialization mode (TAMPTS, and all bits of RTC_TAMPTSCR must be cleared before setting the INIT bit in RTC_ICSR)

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, if TAMPTS = 1 or any bit in TAMP_TAMPTSCR is set, software must either wait for timestamp flag before clearing the tamper flag, or read RTC counter values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx must be disabled and then re-enabled (TAMPxE set to 1) before re-programming backup registers (TAMP_BKPxR). This prevents the application from writing to backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 804. Tamper sampling with precharge pulse

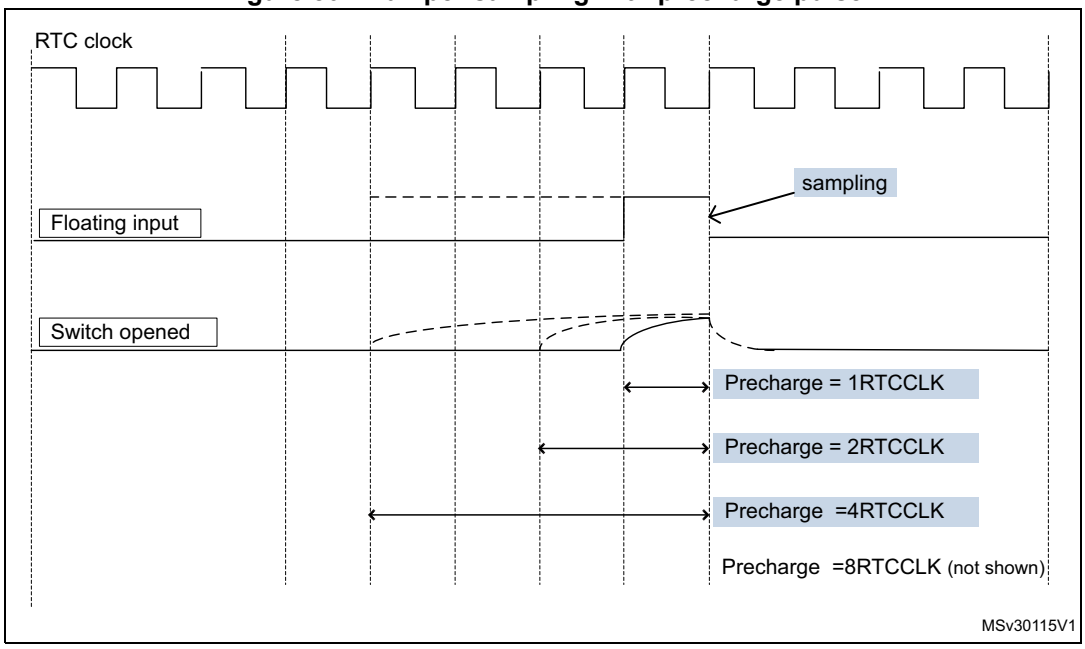
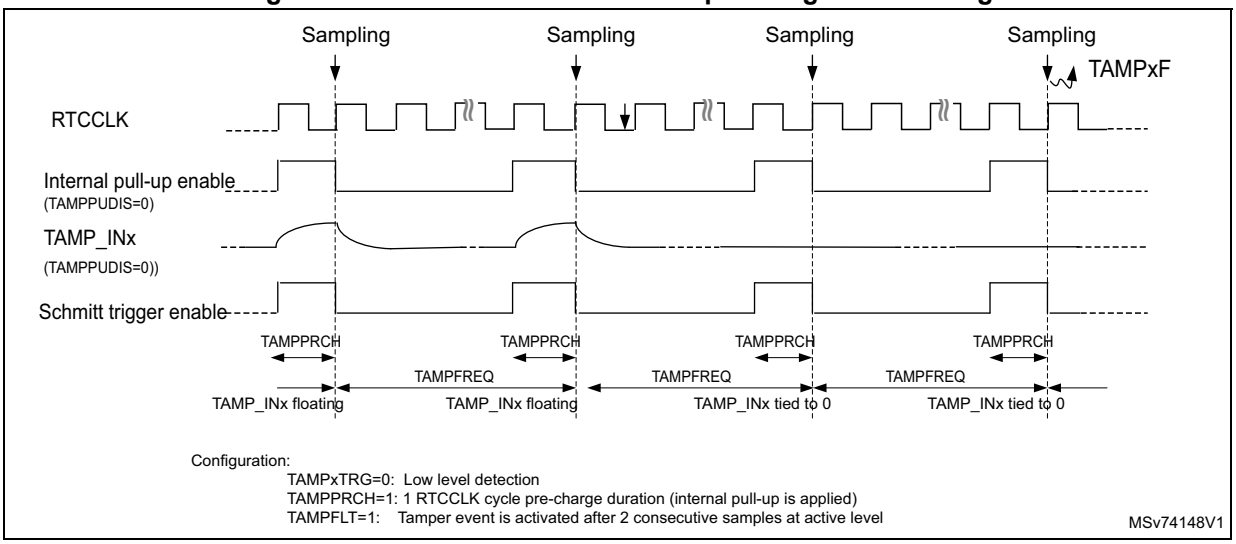


Figure 805. Low level detection with precharge and filtering



Note: Refer to the product datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in TAMP_ATCR1, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0), the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMP_INx (TAMPxEN = TAMPxAM = 1), thanks to ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSEL bits descriptions in TAMP_ATCRx (x = 1, 2) registers.

Every two CK_ATPER cycles (CK_ATPER = 2^{ATPER} × CK_ATPRE), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.

Table 574. Active tamper output change period

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0x0	f_{RTCCLK}	0x0	f_{RTCCLK}	0.030
		0x1	$f_{RTCCLK}/2$	0.061
		0x2	$f_{RTCCLK}/4$	0.122
		0x3	$f_{RTCCLK}/8$	0.244
		0x4	$f_{RTCCLK}/16$	0.488
		0x5	$f_{RTCCLK}/32$	0.977
		0x6	$f_{RTCCLK}/64$	1.953
		0x7	$f_{RTCCLK}/128$	3.906
…	…	…	…	…

Table 574. Active tamper output change period (continued)

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0x7	f_{RTCCLK}/128	0x0	f_{RTCCLK}/128	3.906
		0x1	f_{RTCCLK}/256	7.8125
		0x2	f_{RTCCLK}/512	15.625
		0x3	f_{RTCCLK}/1024	31.250
		0x4	f_{RTCCLK}/2048	62.5
		0x5	f_{RTCCLK}/4096	125
		0x6	f_{RTCCLK}/8192	250
		0x7	f_{RTCCLK}/16384	500
0x8	f_{RTCCLK}/256	0x0	f_{RTCCLK}/256	7.8125
		0x1	f_{RTCCLK}/512	15.625
		0x2	f_{RTCCLK}/1024	31.250
		0x3	f_{RTCCLK}/2048	62.5
		0x4	f_{RTCCLK}/4096	125
		0x5	f_{RTCCLK}/8192	250
		0x6	f_{RTCCLK}/16384	500
		0x7	f_{RTCCLK}/32768	1000
…	…	…	…	…
0xF	f_{RTCCLK}/32768	0x0	f_{RTCCLK}/32768	1000
		0x1	f_{RTCCLK}/65536	2000
		0x2	f_{RTCCLK}/131072	4000
		0x3	f_{RTCCLK}/262144	8000
		0x4	f_{RTCCLK}/524288	16000
		0x5	f_{RTCCLK}/1048576	32000
		0x6	f_{RTCCLK}/2097152	64000
		0x7	f_{RTCCLK}/4194304	128000

1. Assuming f_{RTCCLK} = 32768 Hz.

PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL, and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 575. Minimum ATPER value

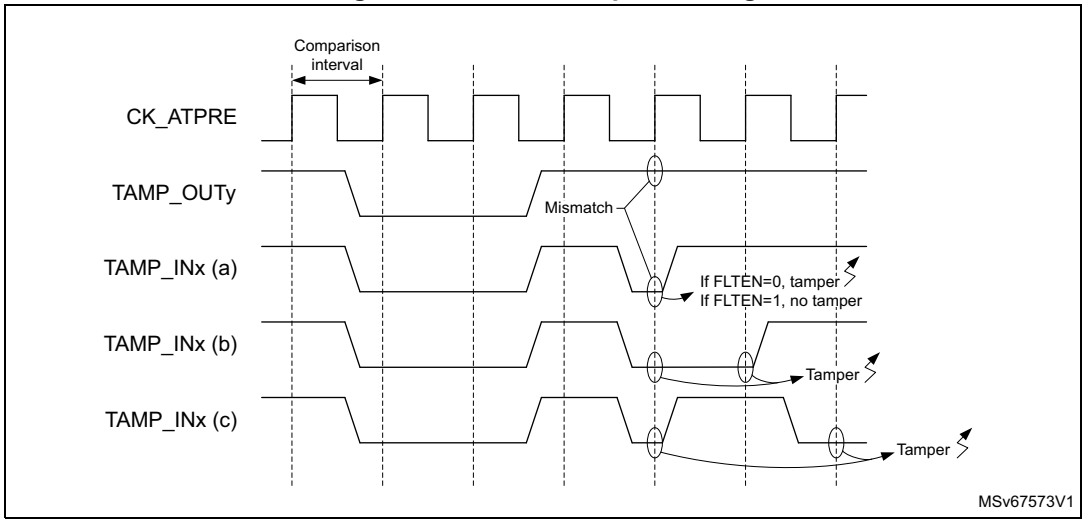
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, every CK_ATPRE cycle. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 000 and ATOSEL2 = 000.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

Figure 806. Active tamper filtering



As illustrated in *Figure 806*, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.

Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in *Table 576* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 576. Active tamper filtered pulse duration

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0x0	f_{RTCCLK}	0.030
0x1	$f_{RTCCLK}/2$	0.061
0x2	$f_{RTCCLK}/4$	0.122
0x3	$f_{RTCCLK}/8$	0.244
0x4	$f_{RTCCLK}/16$	0.488
0x5	$f_{RTCCLK}/32$	0.977
0x6	$f_{RTCCLK}/64$	1.953
0x7	$f_{RTCCLK}/128$	3.906
0x8	$f_{RTCCLK}/256$	7.812
0x9	$f_{RTCCLK}/512$	15.625
0xA	$f_{RTCCLK}/1024$	31.25
0xB	$f_{RTCCLK}/2048$	62.5
0xC	$f_{RTCCLK}/4096$	125
0xD	$f_{RTCCLK}/8192$	250
0xE	$f_{RTCCLK}/16384$	500
0xF	$f_{RTCCLK}/32768$	1000

Note: Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in TAMP_ATSEEDR. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in TAMP_ATOR, and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset.

Read INITS in TAMP_ATOR:

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR1 to configure active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.
- If INITS = 0x1 (initialization already done):

  No initialization. To increase randomness a new SEED must be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode, which switches off the TAMP APB clock.
- If the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

50.4 TAMP low-power modes

Table 577. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Stop mode.
Standby	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Standby mode.

Table 578. TAMP pins functionality over modes

Pin name	Functional in Stop mode	Functional in Standby mode
TAMP_IN[2:1]	Yes	Yes(1)
TAMP_OUT[2:1]	Yes	Yes(2)

1. Only PC13, PA0, PA1 are functional in Standby mode.
2. Only PC13 is functional in Standby mode.

50.5 TAMP interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration (TAMPSEC).

Table 579. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit low-power modes
TAMP	Tamper x(2)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in TAMP_SR.
2. The number of tamper and internal tamper events depend on products.
3. Refer to *Table 577* for more details about available features in the low-power modes.

50.6 TAMP registers

The peripheral registers can be accessed by words (32-bit).

50.6.1 TAMP control register 1 (TAMP_CR1)

Address offset: 0x00

RTC domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 50.3.5*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15E	ITAMP14E	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2E	TAMP1E
														rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15E**: Internal tamper 15 enable
0: Internal tamper 15 disabled.
1: Internal tamper 15 enabled.

Bit 29 **ITAMP14E**: Internal tamper 14 enable
0: Internal tamper 14 disabled.
1: Internal tamper 14 enabled.

Bit 28 **ITAMP13E**: Internal tamper 13 enable
0: Internal tamper 13 disabled.
1: Internal tamper 13 enabled.

Bit 27 **ITAMP12E**: Internal tamper 12 enable
0: Internal tamper 12 disabled.
1: Internal tamper 12 enabled.

Bit 26 **ITAMP11E**: Internal tamper 11 enable
0: Internal tamper 11 disabled.
1: Internal tamper 11 enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
0: Internal tamper 9 disabled.
1: Internal tamper 9 enabled.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 **ITAMP7E**: Internal tamper 7 enable
0: Internal tamper 7 disabled.
1: Internal tamper 7 enabled

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 **ITAMP4E**: Internal tamper 4 enable
0: Internal tamper 4 disabled.
1: Internal tamper 4 enabled.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 **ITAMP2E**: Internal tamper 2 enable
0: Internal tamper 2 disabled.
1: Internal tamper 2 enabled.

Bit 16 **ITAMP1E**: Internal tamper 1 enable
0: Internal tamper 1 disabled.
1: Internal tamper 1 enabled.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)

0: Tamper detection on TAMP_IN2 is disabled.

1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)

0: Tamper detection on TAMP_IN1 is disabled.

1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

50.6.2 TAMP control register 2 (TAMP_CR2)

Address offset: 0x04

RTC domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TAMP2 TRG	TAMP1 TRG	BK ERASE	BK BLOCK	Res.	Res.	Res.	Res.	TAMP2 MSK	TAMP1 MSK
						rw	rw	w	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2 POM	TAMP1 POM
														rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input

0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input

0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.

1: If TAMPFLT ≠ 00 tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase
Writing '1' to this bit reset the backup registers and device secrets(1). Writing 0 has no effect. This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked
0: backup registers and device secrets(1) can be accessed if no tamper flag is set
1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **TAMP2MSK**: Tamper 2 mask
0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.
1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask
0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.
1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **TAMP2POM**: Tamper 2 potential mode
0: Tamper 2 event detection is in confirmed mode(1).
1: Tamper 2 event detection is in potential mode(2).

Bit 0 **TAMP1POM**: Tamper 1 potential mode
0: Tamper 1 event detection is in confirmed mode(1).
1: Tamper 1 event detection is in potential mode(2).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 570: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 570: TAMP interconnection*.

50.6.3 TAMP control register 3 (TAMP_CR3)

Address offset: 0x08

RTC domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ITAMP 15POM	ITAMP 14POM	ITAMP 13POM	ITAMP 12POM	ITAMP 11POM	Res.	ITAMP9 POM	ITAMP8 POM	ITAMP7 POM	ITAMP6 POM	ITAMP5 POM	ITAMP4 POM	ITAMP3 POM	ITAMP2 POM	ITAMP1 POM
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ITAMP15POM**: Internal tamper 15 potential mode
0: Internal tamper 15 event detection is in confirmed mode(1).
1: Internal tamper 15 event detection is in potential mode(2).

Bit 13 **ITAMP14POM**: Internal Tamper 14 potential mode
0: Internal tamper 14 event detection is in confirmed mode(1).
1: Internal tamper 14 event detection is in potential mode(2).

Bit 12 **ITAMP13POM**: Internal tamper 13 potential mode
0: Internal tamper 13 event detection is in confirmed mode(1).
1: Internal tamper 13 event detection is in potential mode(2).

Bit 11 **ITAMP12POM**: Internal tamper 12 potential mode
0: Internal tamper 12 event detection is in confirmed mode(1).
1: Internal tamper 12 event detection is in potential mode(2).

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode(1).
1: Internal tamper 11 event detection is in potential mode(2).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode(1).
1: Internal tamper 9 event detection is in potential mode(2).

Bit 7 **ITAMP8POM**: Internal tamper 8 potential mode
0: Internal tamper 8 event detection is in confirmed mode(1).
1: Internal tamper 8 event detection is in potential mode(2).

Bit 6 **ITAMP7POM**: Internal tamper 7 potential mode
0: Internal tamper 7 event detection is in confirmed mode(1).
1: Internal tamper 7 event detection is in potential mode(2).

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode(1).
1: Internal tamper 4 event detection is in potential mode(2).

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 **ITAMP2POM**: Internal tamper 2 potential mode
0: Internal tamper 2 event detection is in confirmed mode(1).
1: Internal tamper 2 event detection is in potential mode(2).

Bit 0 **ITAMP1POM**: Internal tamper 1 potential mode
0: Internal tamper 1 event detection is in confirmed mode(1).
1: Internal tamper 1 event detection is in potential mode(2).

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 570: TAMP interconnection*
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 570: TAMP interconnection*.

50.6.4 TAMP filter control register (TAMP_FLTCR)

Address offset: 0x0C

RTC domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP PUDIS	TAMPPRCH [1:0]		TAMPFLT [1:0]		TAMPFREQ [2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable
This bit determines if each of the TAMPx pins are precharged before each sample.
0: Precharge TAMP_INx pins before sampling (enable internal pull-up)
1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

50.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

Address offset: 0x10

RTC domain reset value: 0x0007 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTEN	ATO SHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	ATCKSEL[3:0]			
rw	rw				rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2 AM	TAMP1 AM
														rw	rw

Bit 31 **FLTEN**: Active tamper filter enable

0: Active tamper filtering disable

1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30 **ATOSHARE**: Active tamper output sharing

0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi

1: Each active tamper input TAMP_INi is compared with TAMPOUTSELi defined by ATOSELi bits.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period

The tamper output is changed every CK_ATPER = (2^{ATPER} x CK_ATPRE) cycles. Refer to *Table 575: Minimum ATPER value*.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection

These bits selects the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE = RTCCLK/$2^{ATCKSEL}$.

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 CK_ATPRE cycles after all the active tampers are disable.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **TAMP2AM**: Tamper 2 active mode

0: Tamper 2 detection mode is passive.

1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode

0: Tamper 1 detection mode is passive.

1: Tamper 1 detection mode is active.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

50.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

Address offset: 0x14

RTC domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEED[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

50.6.7 TAMP active tamper output register (TAMP_ATOR)

Address offset: 0x18

RTC domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
r	r							r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **INITS**: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.

Bit 14 **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

This field can only be read when the APB is in secure mode.

50.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

Address offset: 0x1C

RTC domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ATOSEL2	Res.	Res.	ATOSEL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw			rw								

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **ATOSEL2**: Active tamper shared output 2 selection

0: TAMPOUTSEL2 = TAMP_OUT1

1: TAMPOUTSEL2 = TAMP_OUT2

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bit 8 **ATOSEL1**: Active tamper shared output 1 selection

0: TAMPOUTSEL1 = TAMP_OUT1

1: TAMPOUTSEL1 = TAMP_OUT2

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 10:9 Reserved, must be kept at reset value.

Bits 7:0 Reserved, must be kept at reset value.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

50.6.9 TAMP secure configuration register (TAMP_SECCFGR)

If TZEN = 1, this register can be written only when the APB access is secure. If TZEN = 0, BKPRWSEC[7:0], BKPWSEC[7:0] and BHKLOCK can be written with nonsecure APB access, and TAMPSEC, CNT1SEC cannot be written.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the TAMP_PRIVCFGR configuration (refer to *Section 50.3.7: TAMP privilege protection modes*).

Address offset: 0x20

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]							
rw	rs							rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPSEC**: Tamper protection (excluding monotonic counters and backup registers)

0: Tamper configuration and interrupt can be written when the APB access is secure or nonsecure.

1: Tamper configuration and interrupt can be written only when the APB access is secure.

Note: Refer to Section 50.3.5: TAMP secure protection modes for details on the read protection.

Bit 30 **BHKLOCK**: Boot hardware key lock

This bit can be read and can only be written to 1 by software. It is cleared by hardware together with the backup registers following a tamper detection event.

0: The Backup registers from TAMP_BKP0R to TAMP_BKP7R can be accessed according to the Protection zone they belong to.

1: The backup registers from TAMP_BKP0R to TAMP_BKP7R cannot be accessed neither in read nor in write (they are read as 0 and write ignore).

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWSEC[7:0]**: Backup registers write protection offset

BKPWSEC value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRWSEC) to TAMP_BKPzR (z = BKPWSEC-1, with BKPWSEC > BKPRWSEC):

– if TZEN=1, these backup registers can be written only with secure access.

– They can be read with secure or nonsecure access.

If BKPWSEC = 0 or if BKPWSEC ≤ BKPRWSEC: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPWSEC if BKPWSEC ≥ BKPRWSEC, else t = BKPRWSEC).

– They can be read or written with secure or nonsecure access.

If BKPWSEC = 32: there is no protection zone 3.

Refer to *Figure 803: Backup register protection zones*.

Note: If TZEN=0: the protection zone 2 can be read and written with nonsecure access.

Note: If BKPWPRIV is set, BKPWSEC[7:0] can be written only in privileged mode.

Bit 15 **CNT1SEC**: Monotonic counter 1 secure protection

0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is secure or nonsecure.

1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is secure.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWSEC[7:0]**: Backup registers read/write protection offset

BKPRWSEC value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWSEC-1, with BKPRWSEC ≥ 1).

– if TZEN=1, these backup registers can be read and written only with secure access.

If BKPRWSEC = 0: there is no protection zone 1.

Refer to *Figure 803: Backup register protection zones*.

Note: If TZEN=0: the protection zone 1 can be read and written with nonsecure access.

Note: If BKPRWPRIV or BKPWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

50.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR)

This register can be written only when the APB access is privileged.

When TZEN = 1, this register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the TAMP_SECCFGR configuration (refer to *Section 50.3.5: TAMP secure protection modes*). BKPWPRIV and BKPRWPRIV can be written in secure mode only.

Address offset: 0x24

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP PRIV	BKP WPRIV	BKPR WPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1 PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding monotonic counter and backup registers)

0: Tamper configuration and interrupt can be written with privileged or unprivileged access.

1: Tamper configuration and interrupt can be written only with privileged access.

Note: Refer to Section 50.3.7: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection

0: Backup registers zone 2 can be written with privileged or unprivileged access.

1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection

0: Backup registers zone 1 can be read and written with privileged or unprivileged access.

1: Backup registers zone 1 can be read and written only with privileged access

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privilege protection

0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is privileged or non-privileged.

1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is privileged.

Bits 14:0 Reserved, must be kept at reset value.

50.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7:*

TAMP privilege protection modes.

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15IE	ITAMP14IE	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2IE	TAMP1IE
														rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15IE**: Internal tamper 15 interrupt enable
0: Internal tamper 15 interrupt disabled.
1: Internal tamper 15 interrupt enabled.

Bit 29 **ITAMP14IE**: Internal tamper 14 interrupt enable
0: Internal tamper 14 interrupt disabled.
1: Internal tamper 14 interrupt enabled.

Bit 28 **ITAMP13IE**: Internal tamper 13 interrupt enable
0: Internal tamper 13 interrupt disabled.
1: Internal tamper 13 interrupt enabled.

Bit 27 **ITAMP12IE**: Internal tamper 12 interrupt enable
0: Internal tamper 12 interrupt disabled.
1: Internal tamper 12 interrupt enabled.

Bit 26 **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22 **ITAMP7IE**: Internal tamper 7 interrupt enable
0: Internal tamper 7 interrupt disabled.
1: Internal tamper 7 interrupt enabled.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16 **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

50.6.12 TAMP status register (TAMP_SR)

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7:*

TAMP privilege protection modes.

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15F	ITAMP14F	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F
	r	r	r	r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2F	TAMP1F
														r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15F**: Internal tamper 15 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 15.

Bit 29 **ITAMP14F**: Internal tamper 14 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 14.

Bit 28 **ITAMP13F**: Internal tamper 13 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 13.

Bit 27 **ITAMP12F**: Internal tamper 12 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 12.

Bit 26 **ITAMP11F**: Internal tamper 11 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9F**: Internal tamper 9 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23 **ITAMP8F**: Internal tamper 8 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 **ITAMP7F**: Internal tamper 7 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21 **ITAMP6F**: Internal tamper 6 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: Internal tamper 4 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **ITAMP2F**: Internal tamper 2 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: Internal tamper 1 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

50.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
	r	r	r	r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2MF	TAMP1MF
														r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15MF**: internal tamper 15 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 15 nonsecure interrupt is raised.

Bit 29 **ITAMP14MF**: internal tamper 14 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 14 nonsecure interrupt is raised.

Bit 28 **ITAMP13MF**: internal tamper 13 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 13 nonsecure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 12 nonsecure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 11 nonsecure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 9 nonsecure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 8 nonsecure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 tamper nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 7 nonsecure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 6 nonsecure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 5 nonsecure interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 4 nonsecure interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 3 nonsecure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 2 nonsecure interrupt is raised.

Bit 16 **ITAMP1MF**:Internal tamper 1 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 1 nonsecure interrupt is raised.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **TAMP2MF**: TAMP2 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 2 nonsecure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 1 nonsecure interrupt is raised.

50.6.14 TAMP secure masked interrupt status register (TAMP_SMISR)

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

Address offset: 0x38

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
	r	r	r	r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2MF	TAMP1MF
														r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15MF**: internal tamper 15 secure interrupt masked flag
This flag is set by hardware when the internal tamper 15 secure interrupt is raised.

Bit 29 **ITAMP14MF**: internal tamper 14 secure interrupt masked flag
This flag is set by hardware when the internal tamper 14 secure interrupt is raised.

Bit 28 **ITAMP13MF**: internal tamper 13 secure interrupt masked flag
This flag is set by hardware when the internal tamper 13 secure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 secure interrupt masked flag
This flag is set by hardware when the internal tamper 12 secure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 secure interrupt masked flag
This flag is set by hardware when the internal tamper 11 secure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 secure interrupt masked flag
This flag is set by hardware when the internal tamper 9 secure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 secure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 secure interrupt masked flag
This flag is set by hardware when the internal tamper 7 secure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 secure interrupt masked flag
This flag is set by hardware when the internal tamper 6 secure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 secure interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 secure interrupt masked flag
This flag is set by hardware when the internal tamper 4 secure interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 secure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 secure interrupt masked flag
This flag is set by hardware when the internal tamper 2 secure interrupt is raised.

Bit 16 **ITAMP1MF**: Internal tamper 1 secure interrupt masked flag
This flag is set by hardware when the internal tamper 1 secure interrupt is raised.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **TAMP2MF**: TAMP2 secure interrupt masked flag
This flag is set by hardware when the tamper 2 secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 secure interrupt masked flag
This flag is set by hardware when the tamper 1 secure interrupt is raised.

50.6.15 TAMP status clear register (TAMP_SCR)

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CITAMP15F	CITAMP14F	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F
	w	w	w	w	w		w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP2F	CTAMP1F
														w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CITAMP15F**: Clear ITAMP15 detection flag
Writing 1 in this bit clears the ITAMP15F bit in the TAMP_SR register.

Bit 29 **CITAMP14F**: Clear ITAMP14 detection flag
Writing 1 in this bit clears the ITAMP14F bit in the TAMP_SR register.

Bit 28 **CITAMP13F**: Clear ITAMP13 detection flag
Writing 1 in this bit clears the ITAMP13F bit in the TAMP_SR register.

Bit 27 **CITAMP12F**: Clear ITAMP12 detection flag
Writing 1 in this bit clears the ITAMP12F bit in the TAMP_SR register.

Bit 26 **CITAMP11F**: Clear ITAMP11 detection flag
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **CITAMP9F**: Clear ITAMP9 detection flag
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 **CITAMP7F**: Clear ITAMP7 detection flag
Writing 1 in this bit clears the ITAMP7F bit in the TAMP_SR register.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

50.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

Address offset: 0x040

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COUNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **COUNT[31:0]:**

The monotonic counter increments by one when a write access to this register occurs, regardless of the written value. Reading this register returns the monotonic counter value. This register does not roll-over and freezes upon reaching its maximum value.

50.6.17 TAMP option register (TAMP_OR)

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

Address offset: 0x50

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT2_RMP	Res.	OUT1_RMP	Res.	IN2_RMP	Res.	IN1_RMP
									rw		rw		rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **OUT2_RMP**: TAMP_OUT2 mapping
0x0: TAMP_OUT2 is mapped on PB9
0x1: TAMP_OUT2 is mapped on PE3

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OUT1_RMP**: TAMP_OUT1 mapping
0x0: TAMP_OUT1 is mapped on PA2
0x1: TAMP_OUT1 is mapped on PC13

Bit 3 Reserved, must be kept at reset value.

Bit 2 **IN2_RMP**: TAMP_IN2 mapping
0x0: TAMP_IN2 is mapped on PA1
0x1: TAMP_IN2 is mapped on PC13

Bit 1 Reserved, must be kept at reset value.

Bit 0 **IN1_RMP**: TAMP_IN1 mapping
0x0: TAMP_IN1 is mapped on PA0
0x1: TAMP_IN1 is mapped on PB9

50.6.18 TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

Address offset: 0x54

RTC domain reset value: 0x8000 0000 (reset also by RTCDRST bit in RCC_RTCCR register)

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RP CFG31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RP CFG8	RP CFG7	RP CFG6	RP CFG5	RP CFG4	RP CFG3	RP CFG2	RP CFG1	Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

Bit 31 **RPCFG31**: Backup registers protection
0: Backup registers and BHKLOCK are not included in the device secrets protected by TAMP peripheral
1: Backup registers and BHKLOCK are included in the device secrets protected by TAMP peripheral

Bits 30:9 Reserved, must be kept at reset value.

Bit 8 **RPCFG8**: Configurable resource 8 protection(1)
0: Resource 8 is not included in the device secrets protected by TAMP peripheral
1: Resource 8 is included in the device secrets protected by TAMP peripheral

Bit 7 **RPCFG7**: Configurable resource 7 protection(2)
0: Resource 7 is not included in the device secrets protected by TAMP peripheral
1: Resource 7 is included in the device secrets protected by TAMP peripheral

Bit 6 **RPCFG6**: Configurable resource 6 protection(3)
0: Resource 6 is not included in the device secrets protected by TAMP peripheral
1: Resource 6 is included in the device secrets protected by TAMP peripheral

Bit 5 **RPCFG5**: Configurable resource 5 protection(4)
0: Resource 5 is not included in the device secrets protected by TAMP peripheral
1: Resource 5 is included in the device secrets protected by TAMP peripheral

Bit 4 **RPCFG4**: Configurable resource 4 protection(5)
0: Resource 4 is not included in the device secrets protected by TAMP peripheral
1: Resource 4 is included in the device secrets protected by TAMP peripheral

Bit 3 **RPCFG3**: Configurable resource 3 protection(6)
0: Resource 3 is not included in the device secrets protected by TAMP peripheral
1: Resource 3 is included in the device secrets protected by TAMP peripheral

Bit 2 **RPCFG2**: Configurable resource 2 protection(7)
0: Resource 2 is not included in the device secrets protected by TAMP peripheral
1: Resource 2 is included in the device secrets protected by TAMP peripheral

Bit 1 **RPCFG1**: Configurable resource 1 protection(8)
0: Resource 1 is not included in the device secrets protected by TAMP peripheral
1: Resource 1 is included in the device secrets protected by TAMP peripheral

Bit 0 Reserved, must be kept at reset value.

1. Refer to tamp_confirmed_rpcfg8 and tamp_potential_rpcfg8 signals in *Table 568: TAMP input/output pins* and *Table 570: TAMP interconnection*.
2. Refer to tamp_confirmed_rpcfg7 and tamp_potential_rpcfg7 signals in *Table 568: TAMP input/output pins* and *Table 570: TAMP interconnection*.
3. Refer to tamp_confirmed_rpcfg6 and tamp_potential_rpcfg6 signals in *Table 568: TAMP input/output pins* and *Table 570: TAMP interconnection*.
4. Refer to tamp_confirmed_rpcfg5 and tamp_potential_rpcfg5 signals in *Table 568: TAMP input/output pins* and *Table 570: TAMP interconnection*.
5. Refer to tamp_confirmed_rpcfg4 and tamp_potential_rpcfg4 signals in *Table 568: TAMP input/output pins* and *Table 570: TAMP interconnection*.
6. Refer to tamp_confirmed_rpcfg3 and tamp_potential_rpcfg3 signals in *Table 568: TAMP input/output pins* and *Table 570: TAMP interconnection*.
7. Refer to tamp_confirmed_rpcfg2 and tamp_potential_rpcfg2 signals in *Table 568: TAMP input/output pins* and *Table 570: TAMP interconnection*.
8. Refer to tamp_confirmed_rpcfg1 and tamp_potential_rpcfg1 signals in *Table 568: TAMP input/output pins* and *Table 570: TAMP interconnection*.

50.6.19 TAMP backup x register (TAMP_BKPxR)

This register can be protected against nonsecure access. Refer to *Section 50.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 50.3.7: TAMP privilege protection modes*.

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 **BKP[31:0]:**

The application can write or read data to and from these registers.

In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set.

50.6.20 TAMP register map

Table 580. TAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TAMP_CR1	Res.	ITAMP15E	ITAMP14E	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2E	TAMP1E
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0															0	0
0x04	TAMP_CR2	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	Res.	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2POM	TAMP1POM
	Reset value							0	0	0	0					0	0															0	0
0x08	TAMP_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP15POM	ITAMP14POM	ITAMP13POM	ITAMP12POM	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	ITAMP4POM	ITAMP3POM	ITAMP2POM	ITAMP1POM
	Reset value																		0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]			TAMPFREQ[2:0]		
	Reset value																								0	0	0	0	0	0	0	0	0
0x10	TAMP_ATCR1	FLTEN	ATOSHARE	Res.	Res.	Res.	AT PER[2:0]			Res.	Res.	Res.	Res.	ATCK SEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0					0	1	1	1															0	0
0x14	TAMP_ ATSEEDR	SEED[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	TAMP_ATOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
	Reset value																	0	0							0	0	0	0	0	0	0	0
0x1C	TAMP_ATCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATOSEL2	Res.	Res.	ATOSEL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																					0			0								
0x20	TAMP_SEC CFGR	TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]								CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]								
	Reset value	0	0							0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0
0x24	TAMP_ PRIVCFGR	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT1PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0															

Table 580. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TAMP_IER	Res.	ITAMP15IE	ITAMP14IE	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IF	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2IE	TAMP1IE
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0															0	0
0x30	TAMP_SR	Res.	ITAMP15F	ITAMP14F	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2F	TAMP1F
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0															0	0
0x34	TAMP_MISR	Res.	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2MF	TAMP1MF
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0															0	0
0x38	TAMP_SMISR	Res.	ITAMP15MF	ITAMP14MF	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP2MF	TAMP1MF
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0															0	0
0x3C	TAMP_SCR	Res.	CITAMP15F	CITAMP14F	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP2F	CTAMP1F
	Reset value		0	0	0	0	0		0	0	0	0	0	0	0	0	0															0	0
0x40	TAMP_ COUNT1R	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TAMP_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT2_RMP	Res.	OUT1_RMP	Res.	IN2_RMP	Res.	IN1_RMP
	Reset value																										0		0		0		0
0x54	TAMP_ RPCFGR	RPCFG31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFG8	RPCFG7	RPCFG6	RPCFG5	RPCFG4	RPCFG3	RPCFG2	RPCFG1	Res.
	Reset value	1																							0	0	0	0	0	0	0	0	
0x100 + 0x04*x, (x= 0 to 31)	TAMP_BKPxR	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

51 Inter-integrated circuit interface (I2C)

51.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

51.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.3 compatibility[(a)]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.5 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to *Section 51.3: I2C implementation*.

51.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 581. I2C implementation

I2C features(1)	I2C1	I2C2	I2C3	I2C4
7-bit addressing mode	X	X	X	X
10-bit addressing mode	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X
Wake-up from Stop mode only (no autonomous mode)	X	X	X	X
SMBus/PMBus	X	X	X	X
SMBus 3.3 VIL/VIH support	X	X	X	X

1. X = supported.

When initializing I2C for using the device on an SMBus 3.3 compliant bus, it is recommended to enable the corresponding I2Cx_SMB3EN bit of the SBS_PMCR register. This shifts the V_{IH} threshold of the corresponding GPIO (configured as GPIO AF) to a lower level.

51.4 I2C functional description

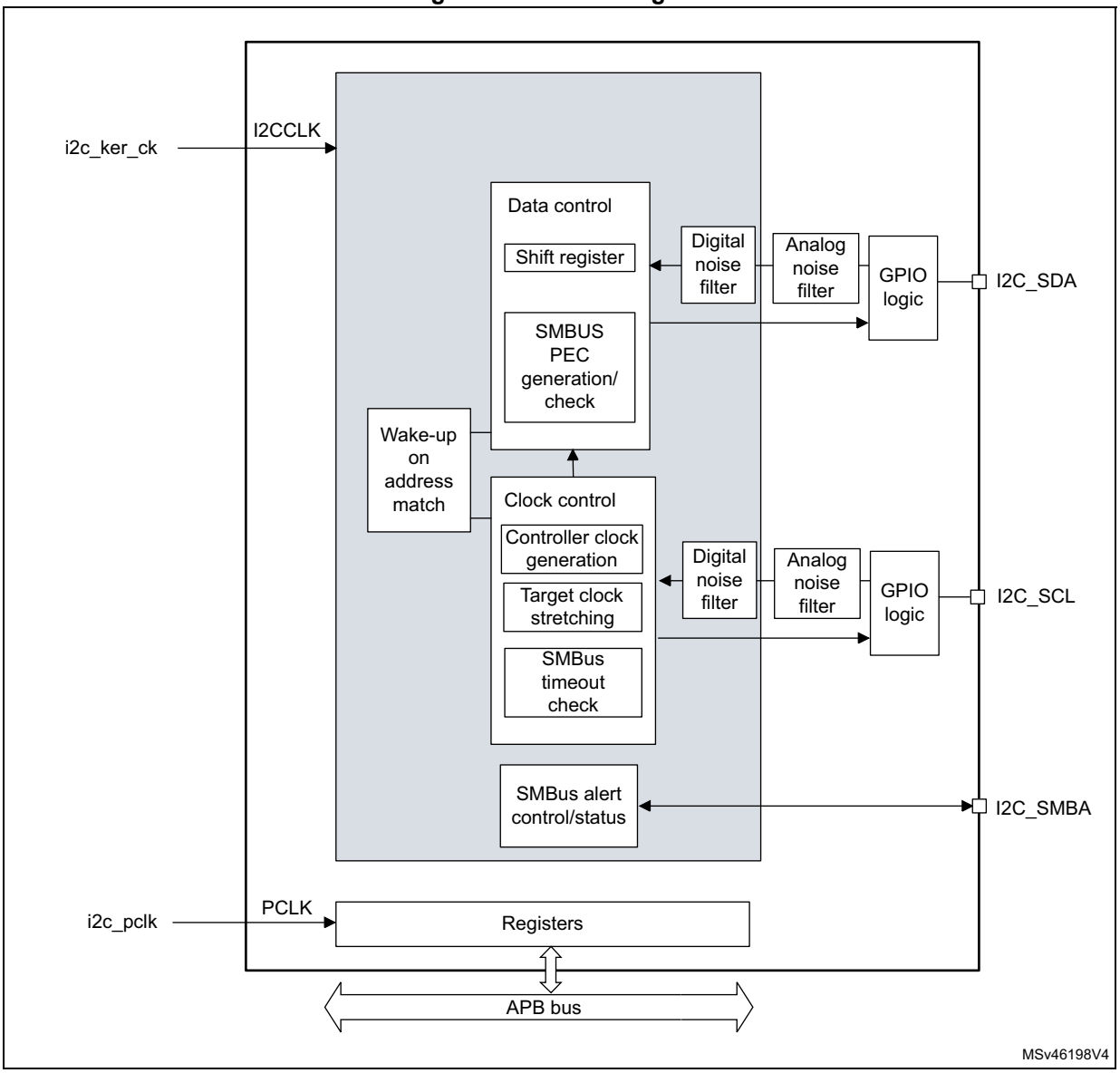
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

51.4.1 I2C block diagram

Figure 807. Block diagram



51.4.2 I2C pins and internal signals

Table 582. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 583. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 596* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

51.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

51.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

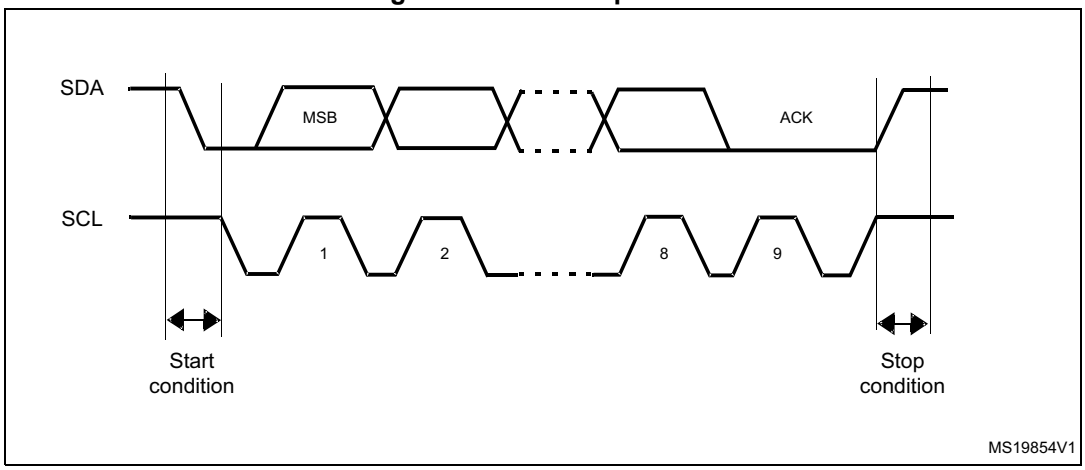
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 808. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

51.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 51.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 584. Comparison of analog and digital filters

Item	Analog filter	Digital filter
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Wake-up from Stop mode on address match not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

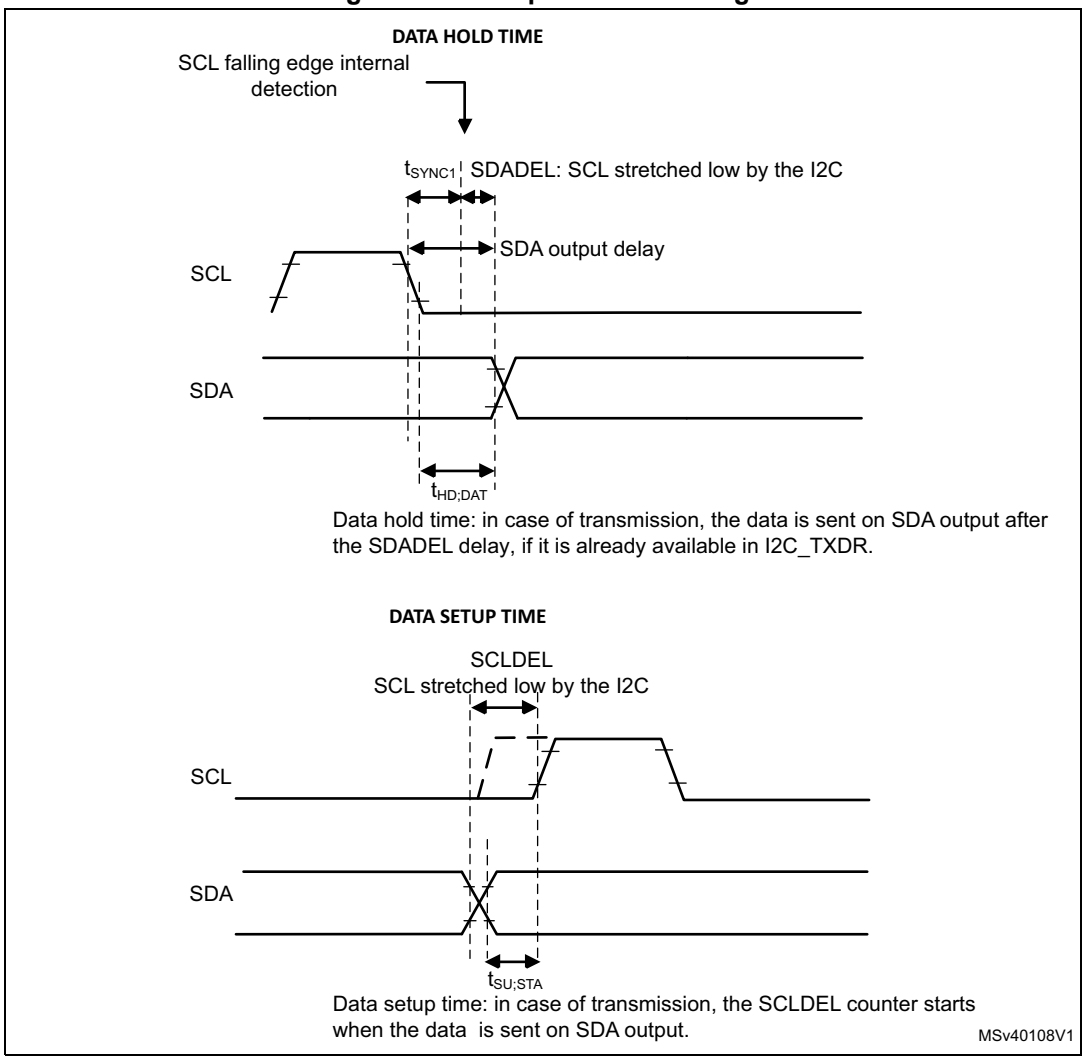
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 809. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

{$t_{f(max)}$ + $t_{HD;DAT(min)}$ - $t_{AF(min)}$ - [(DNF + 3) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}} ≤ SDADEL

SDADEL ≤ {$t_{HD;DAT\ (max)}$ - $t_{AF(max)}$ - [(DNF + 4) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}}

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 585. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

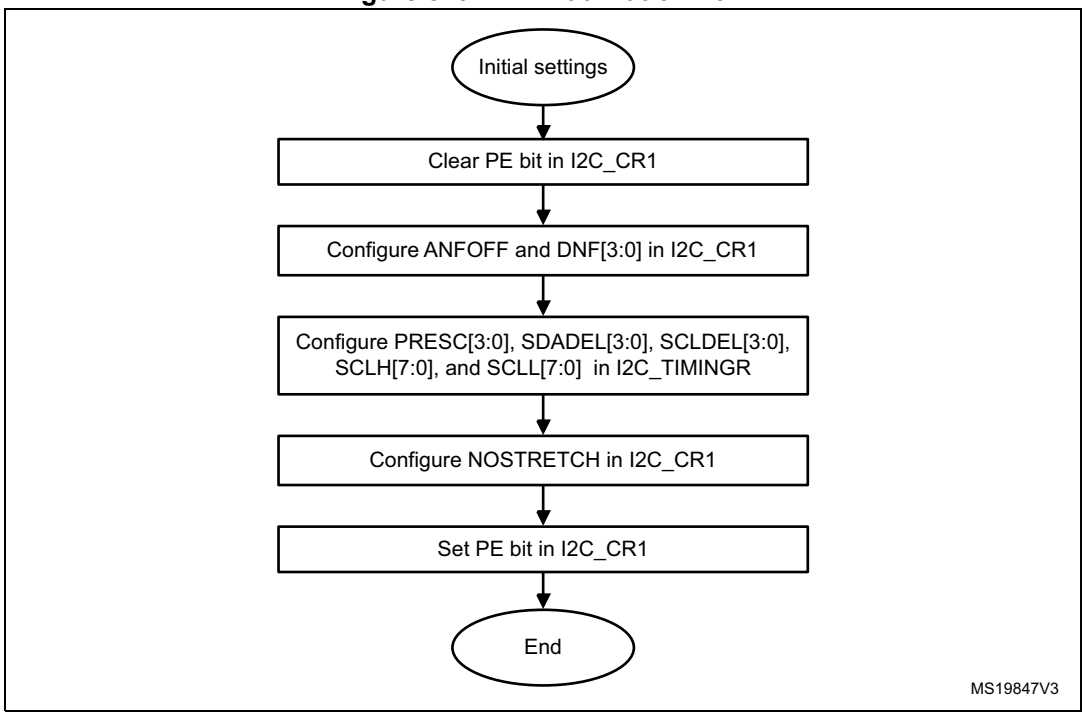
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 810. I2C initialization flow



51.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

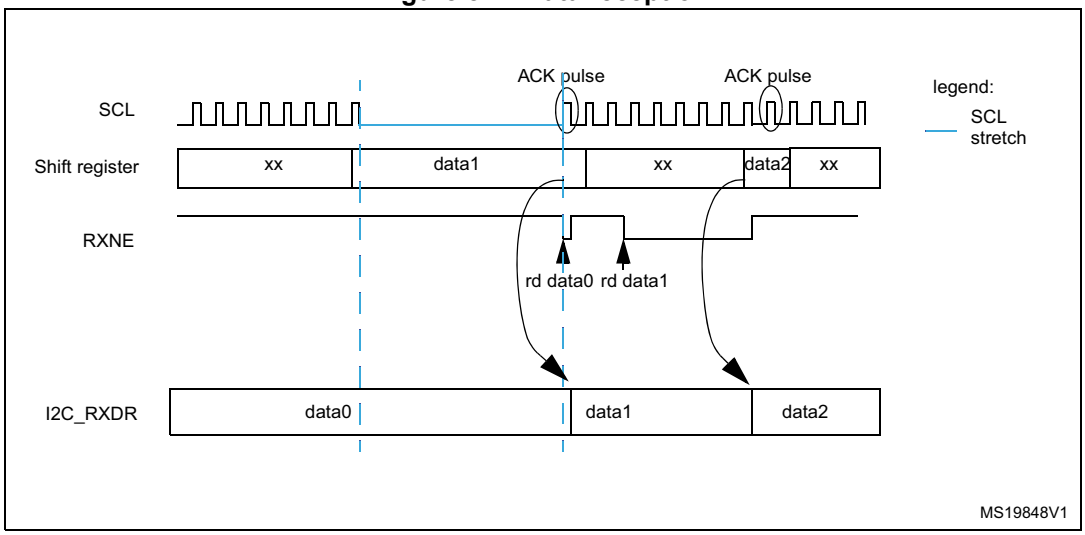
51.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

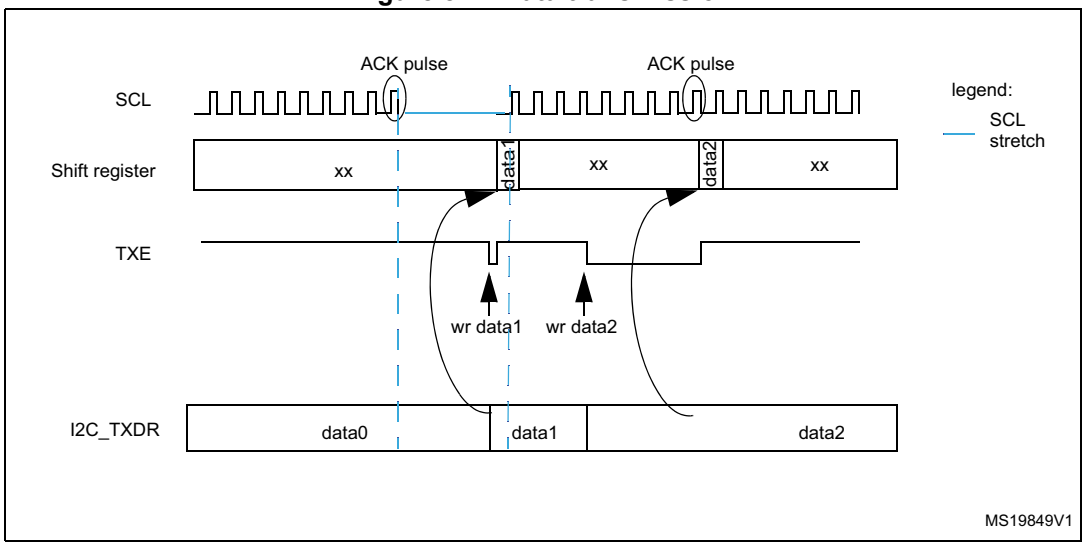
Figure 811. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 812. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 586. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

51.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not

support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

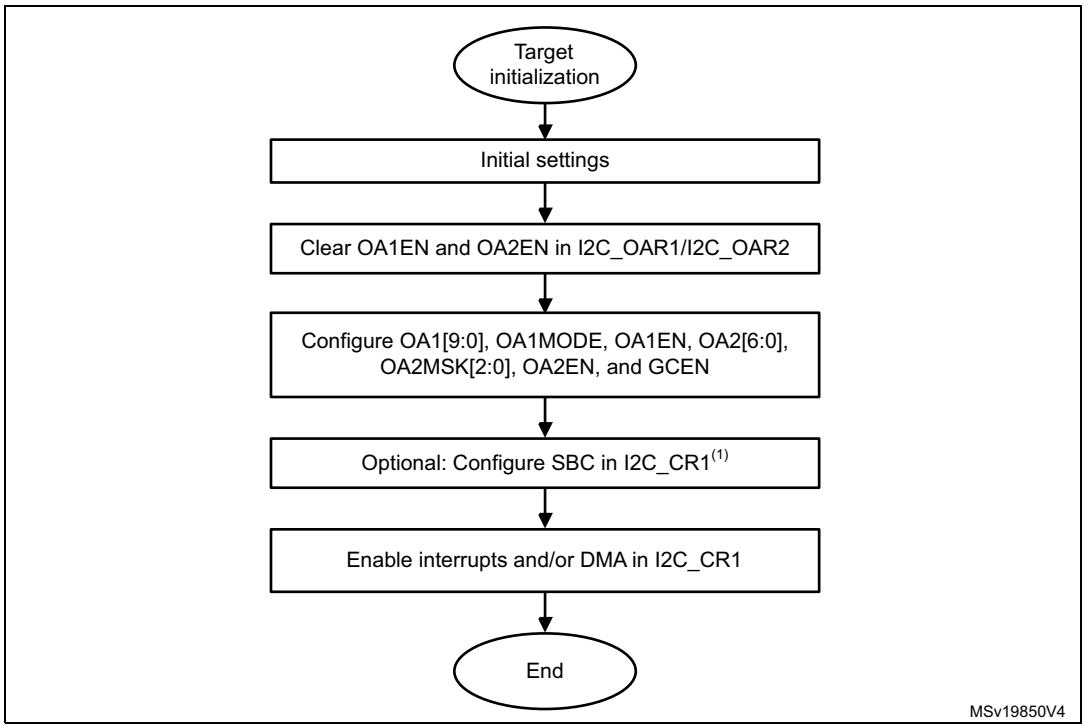
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 813. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 814. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

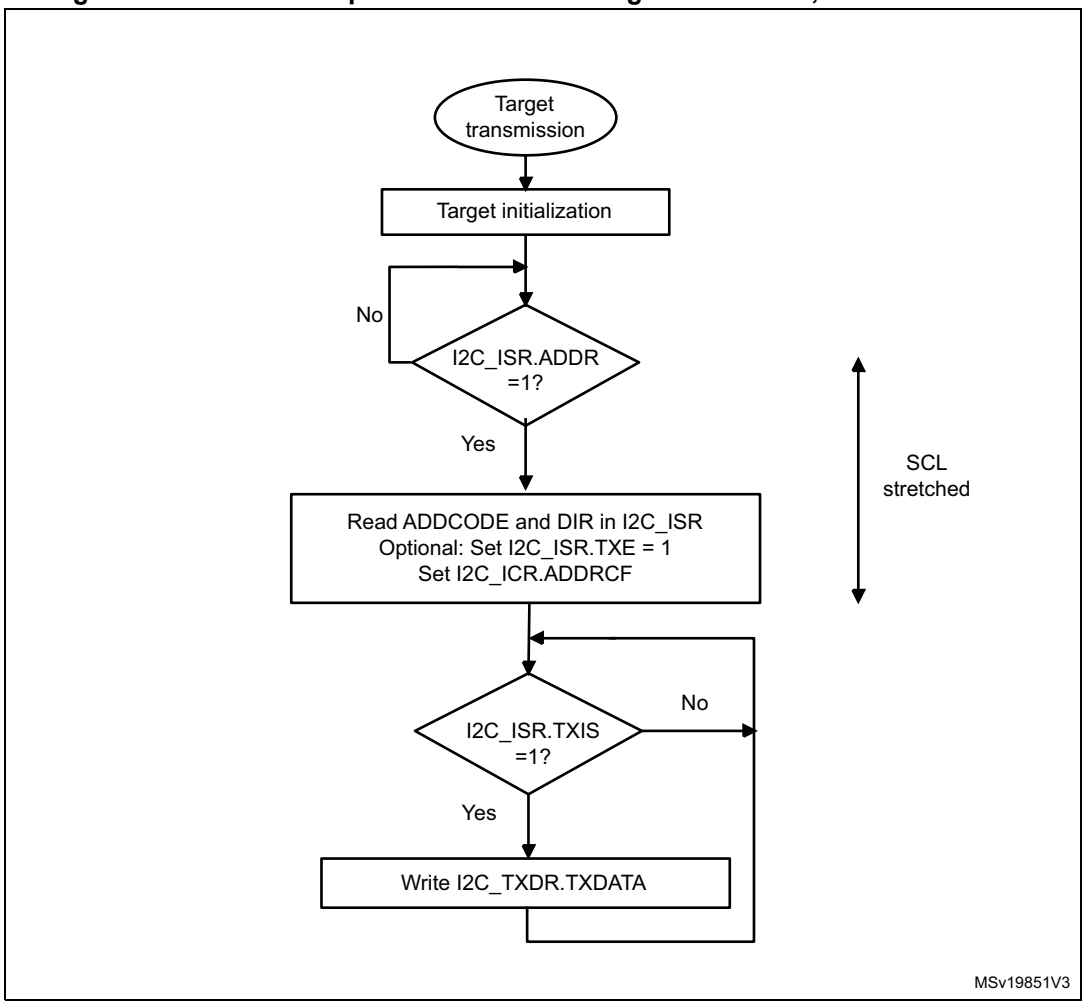


Figure 815. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

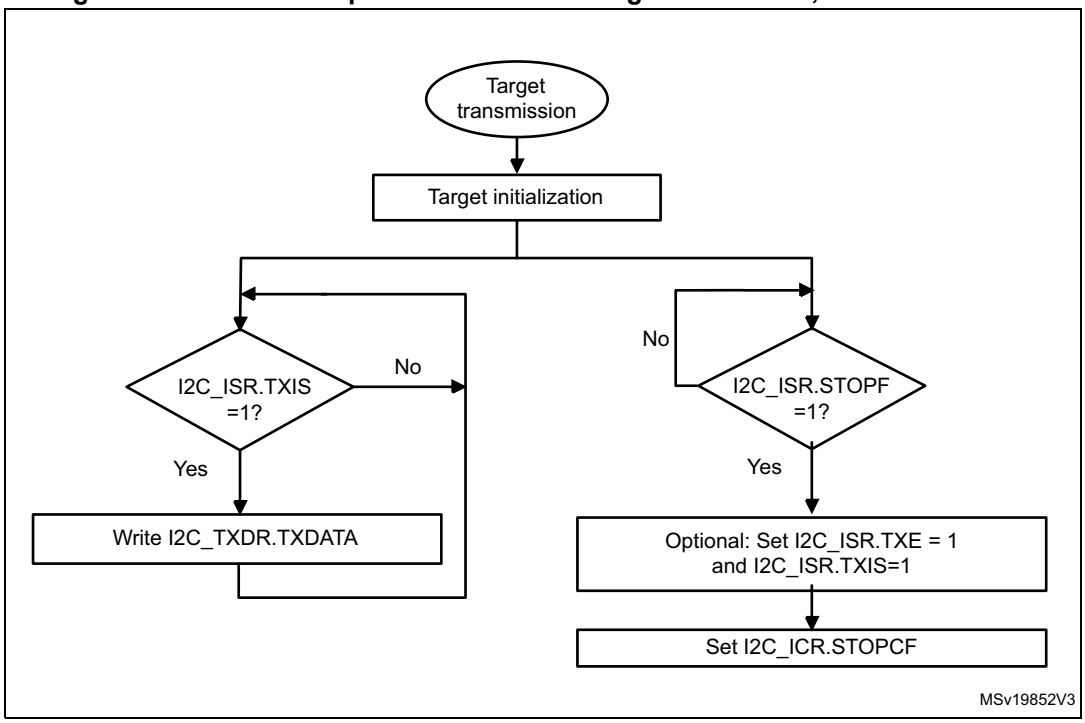
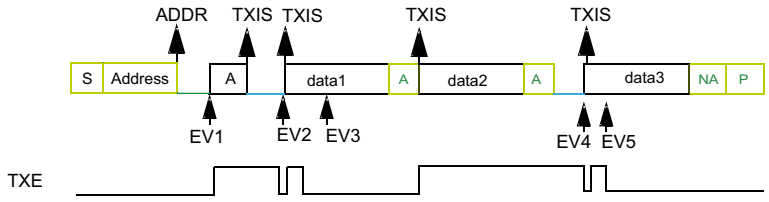


Figure 816. Transfer bus diagrams for I2C target transmitter (mandatory events only)

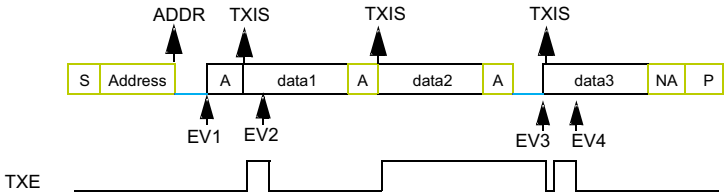
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

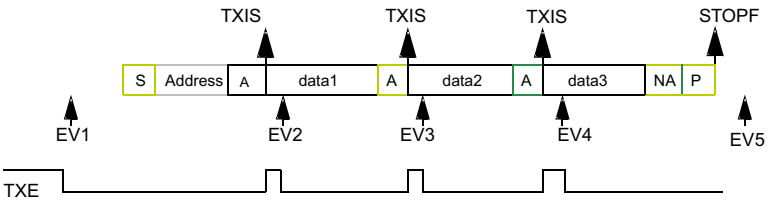
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



legend:
transmission
reception
SCL stretch

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 817. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

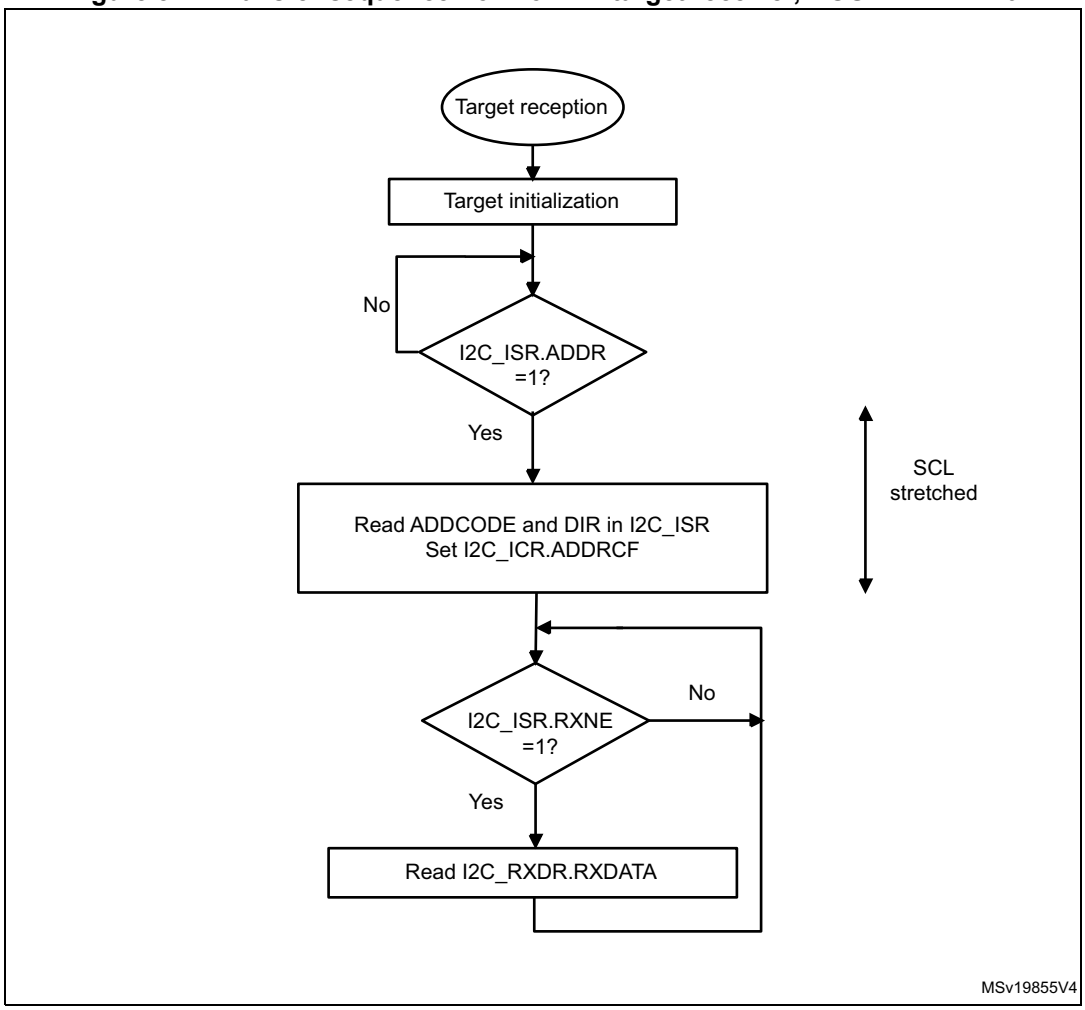


Figure 818. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

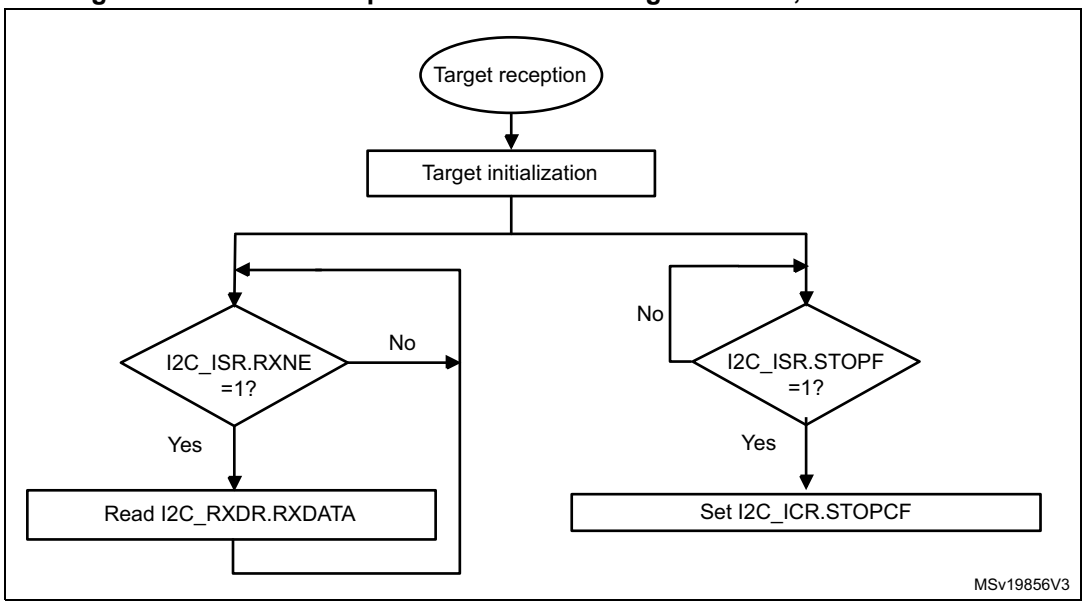
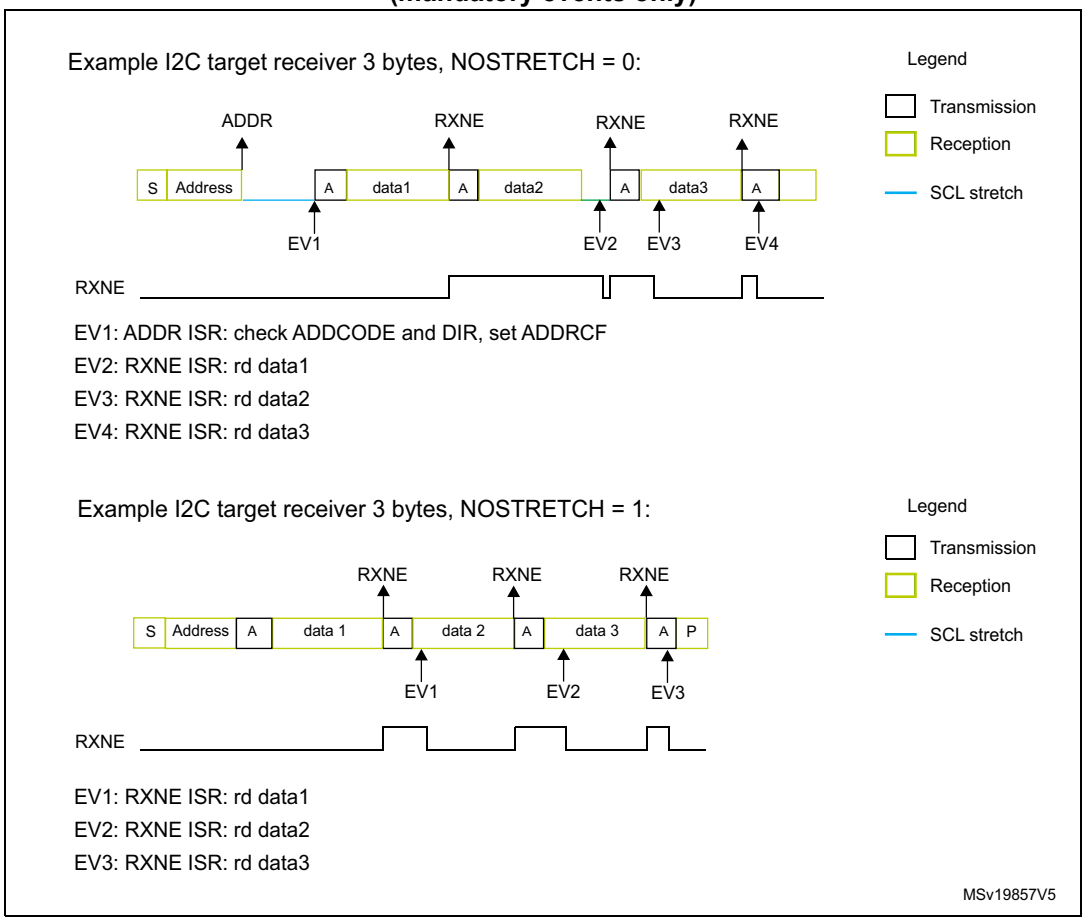


Figure 819. Transfer bus diagrams for I2C target receiver (mandatory events only)



51.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

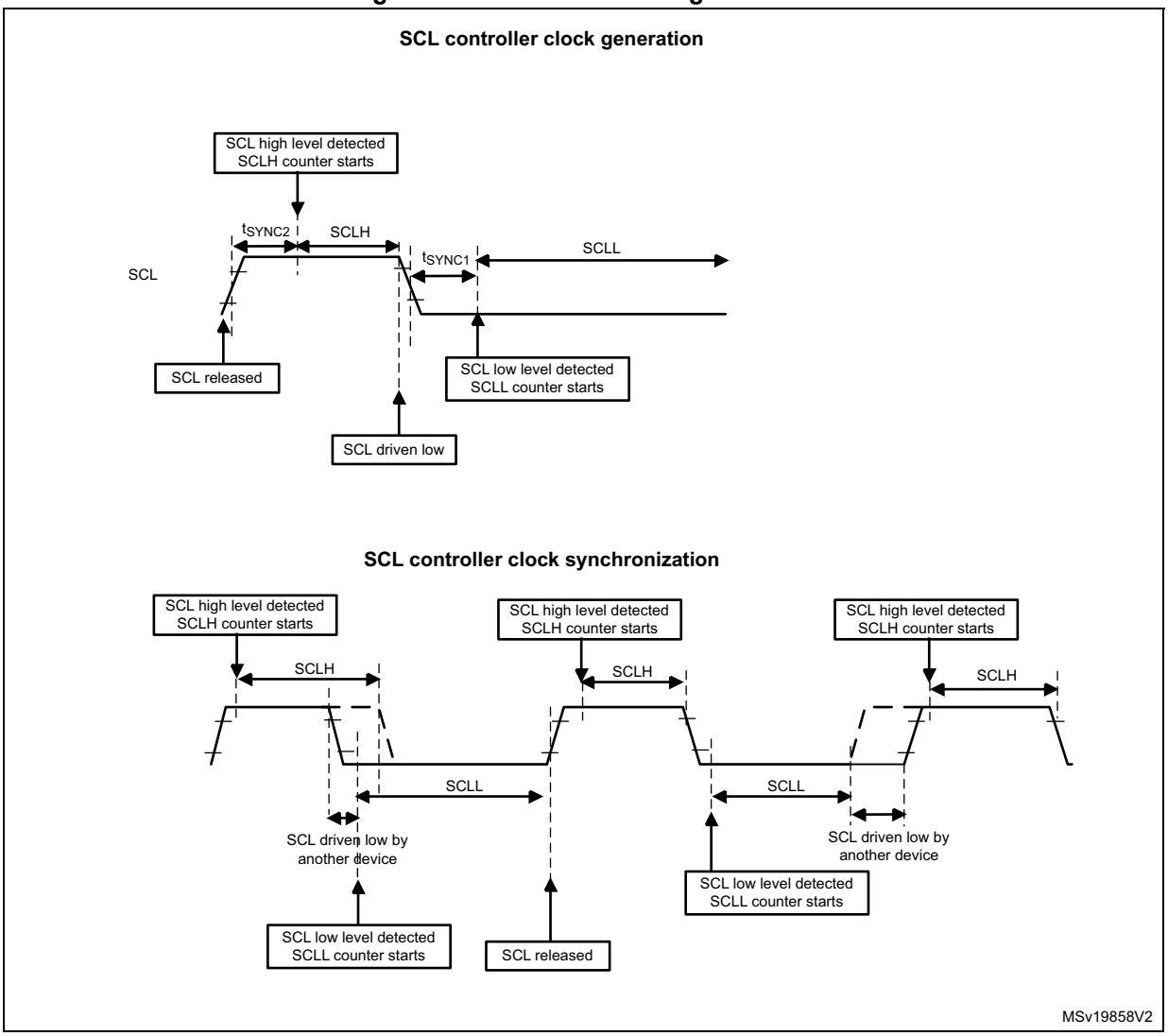
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 820. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 587. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 51.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

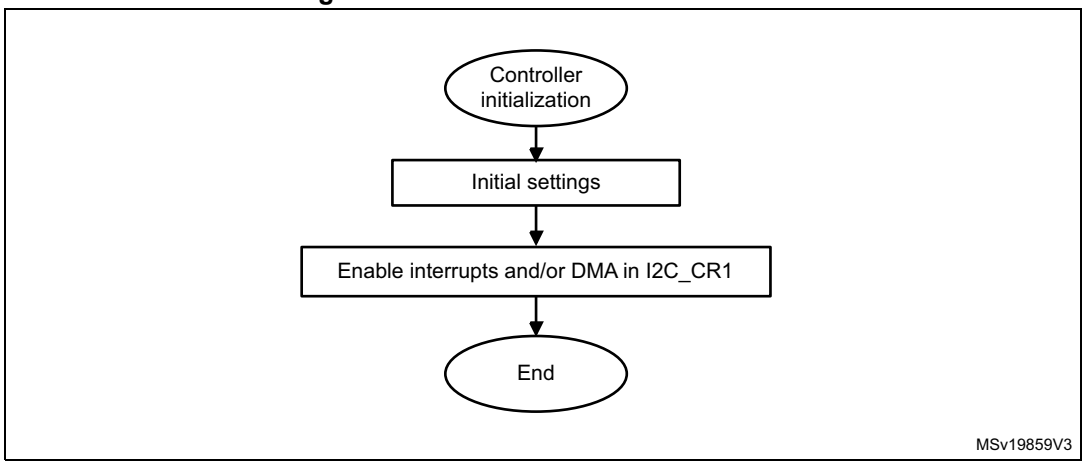
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 821. Controller initialization flow

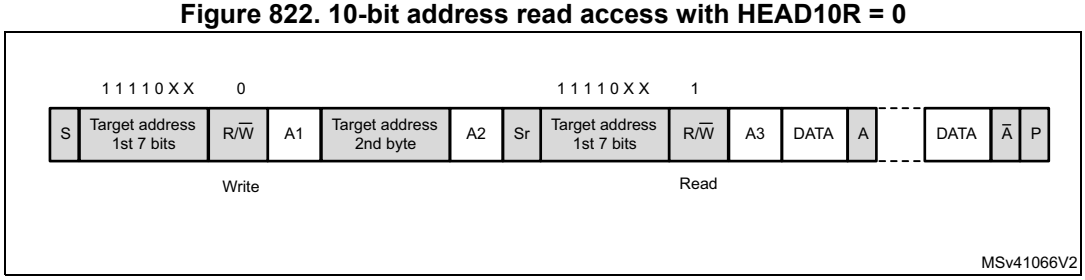


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

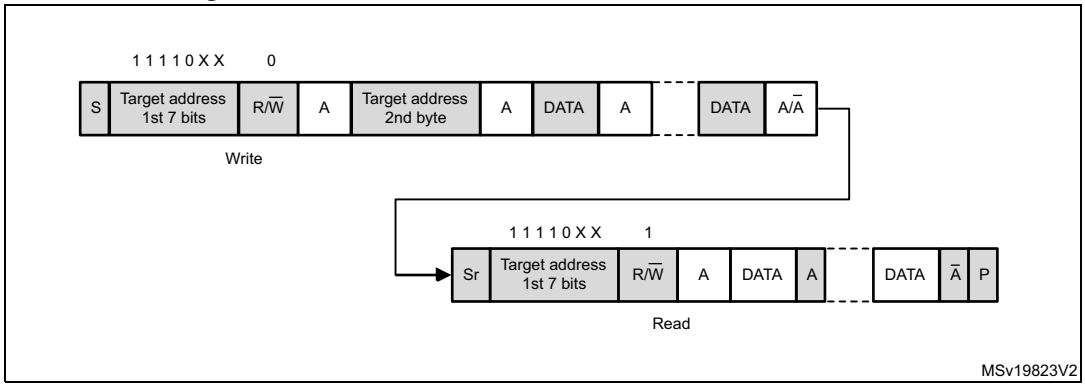
Figure 822. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 823. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 824. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

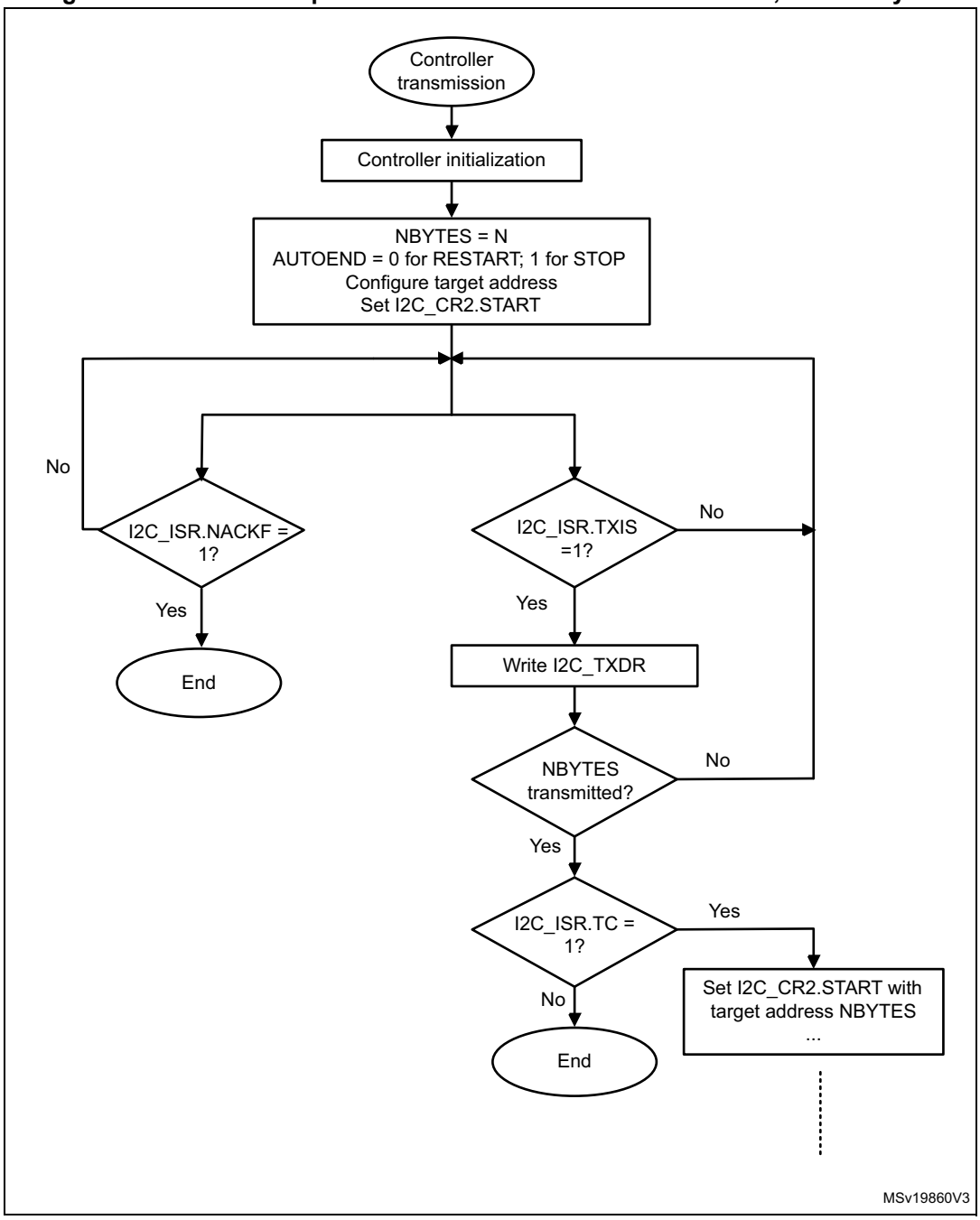


Figure 825. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

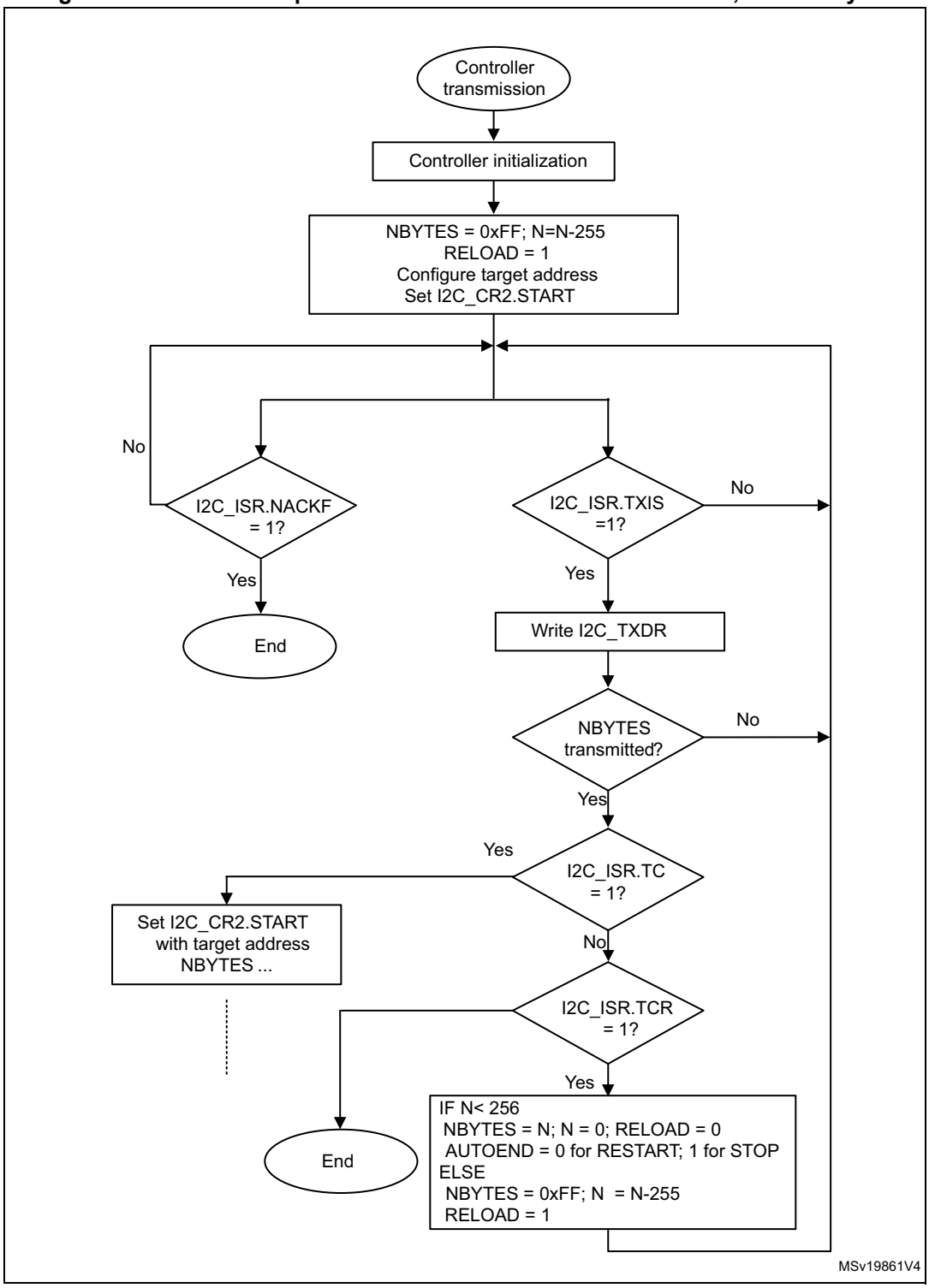
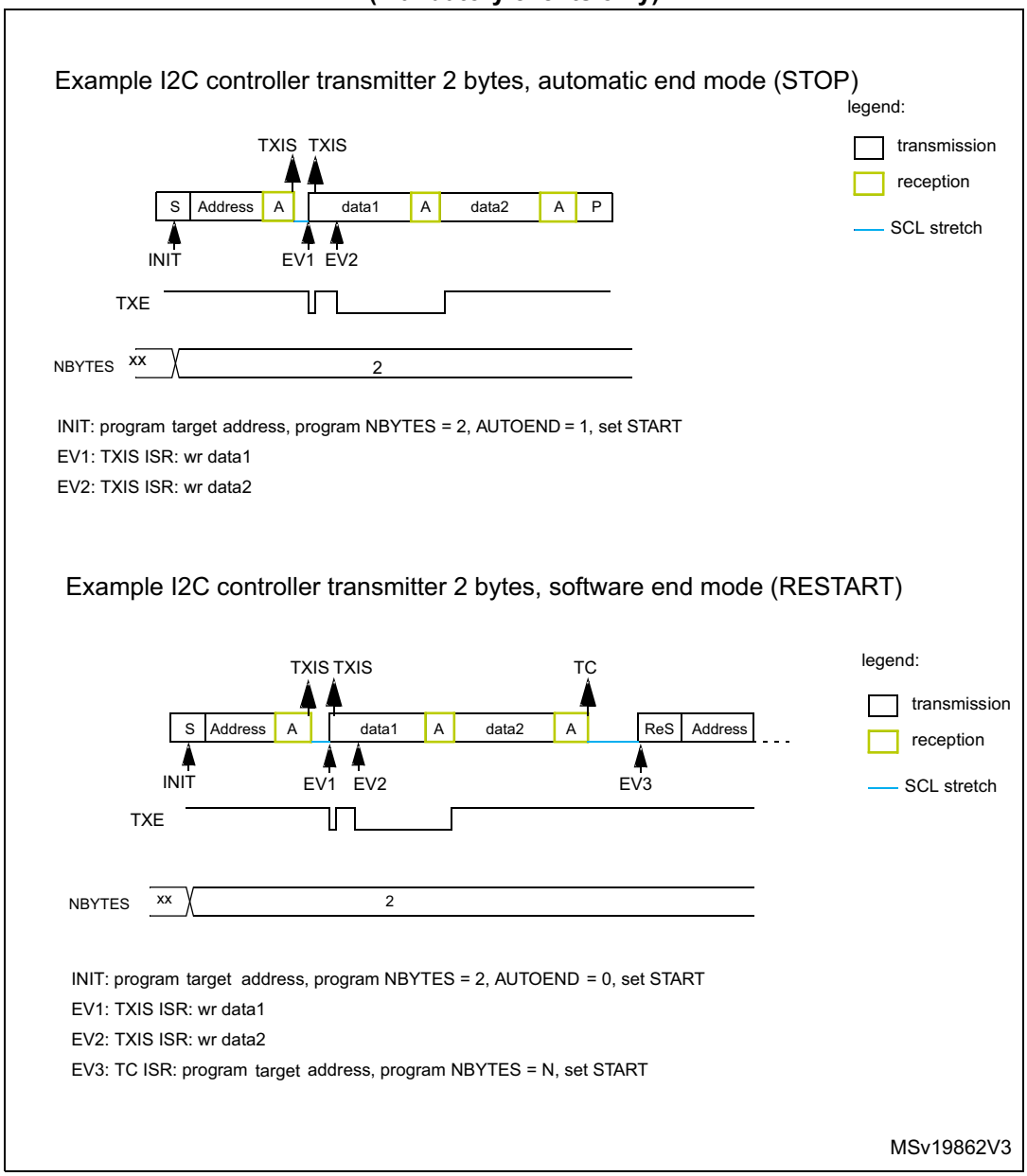


Figure 826. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 827. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

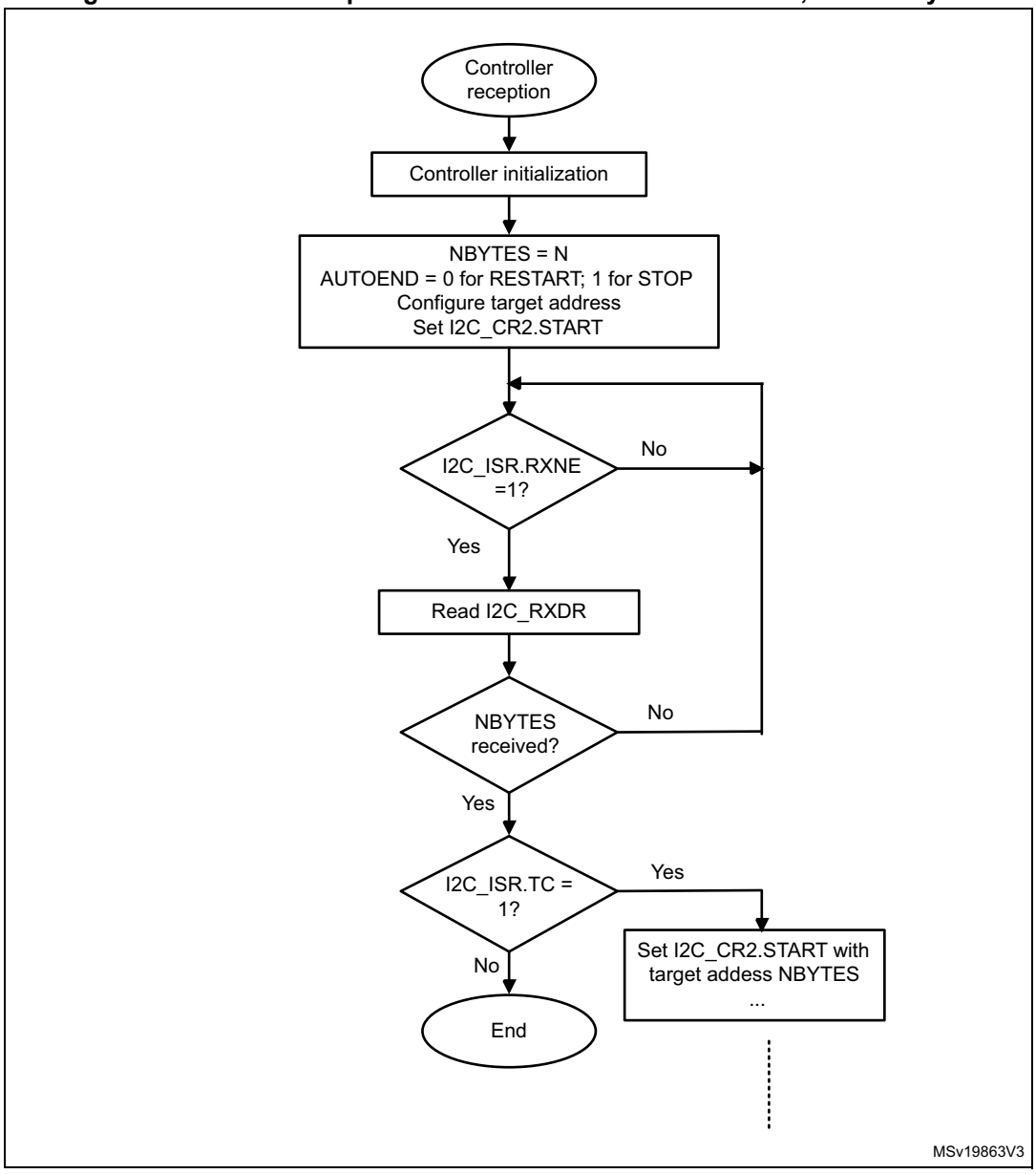


Figure 828. Transfer sequence flow for I2C controller receiver, N > 255 bytes

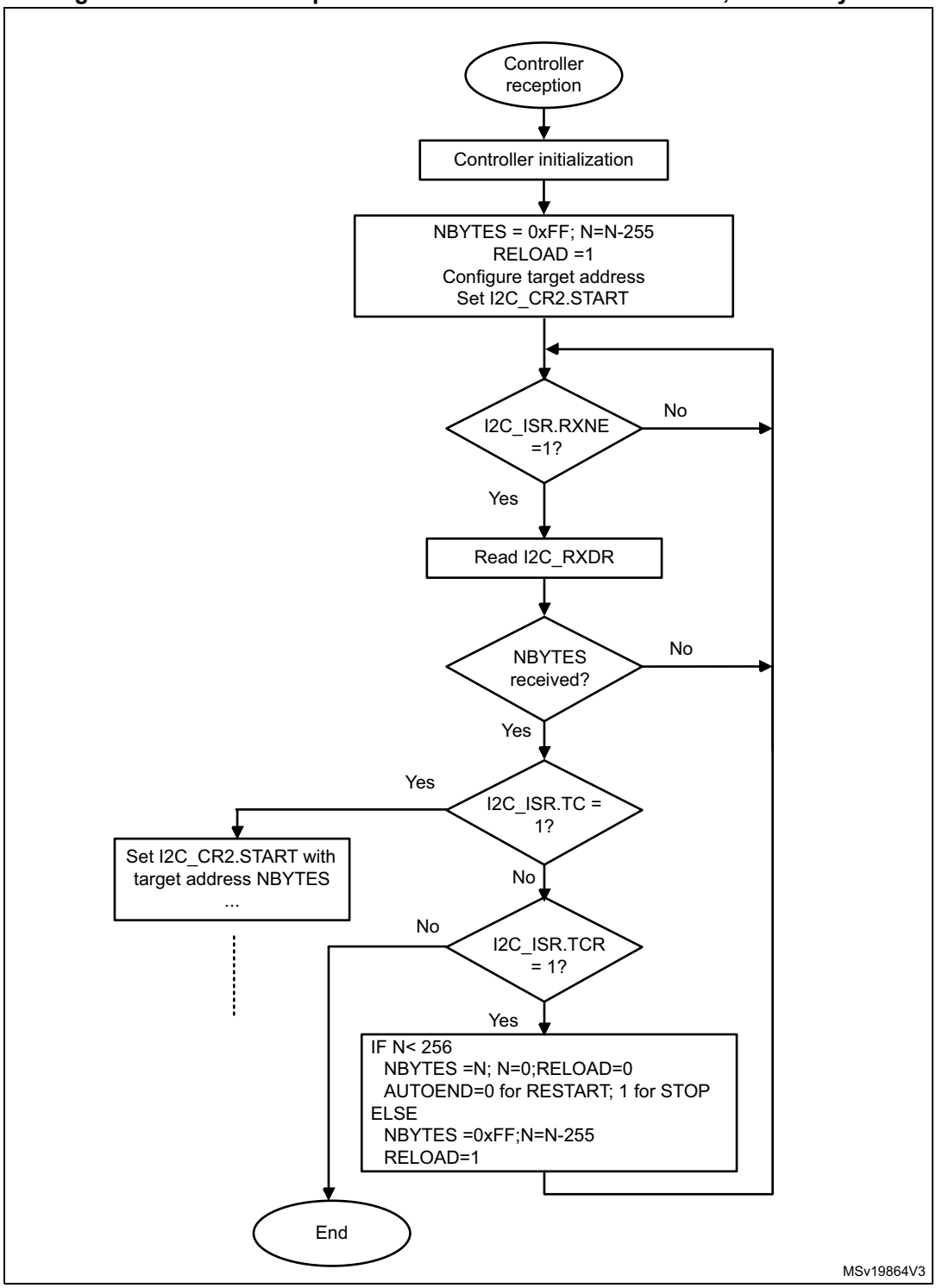
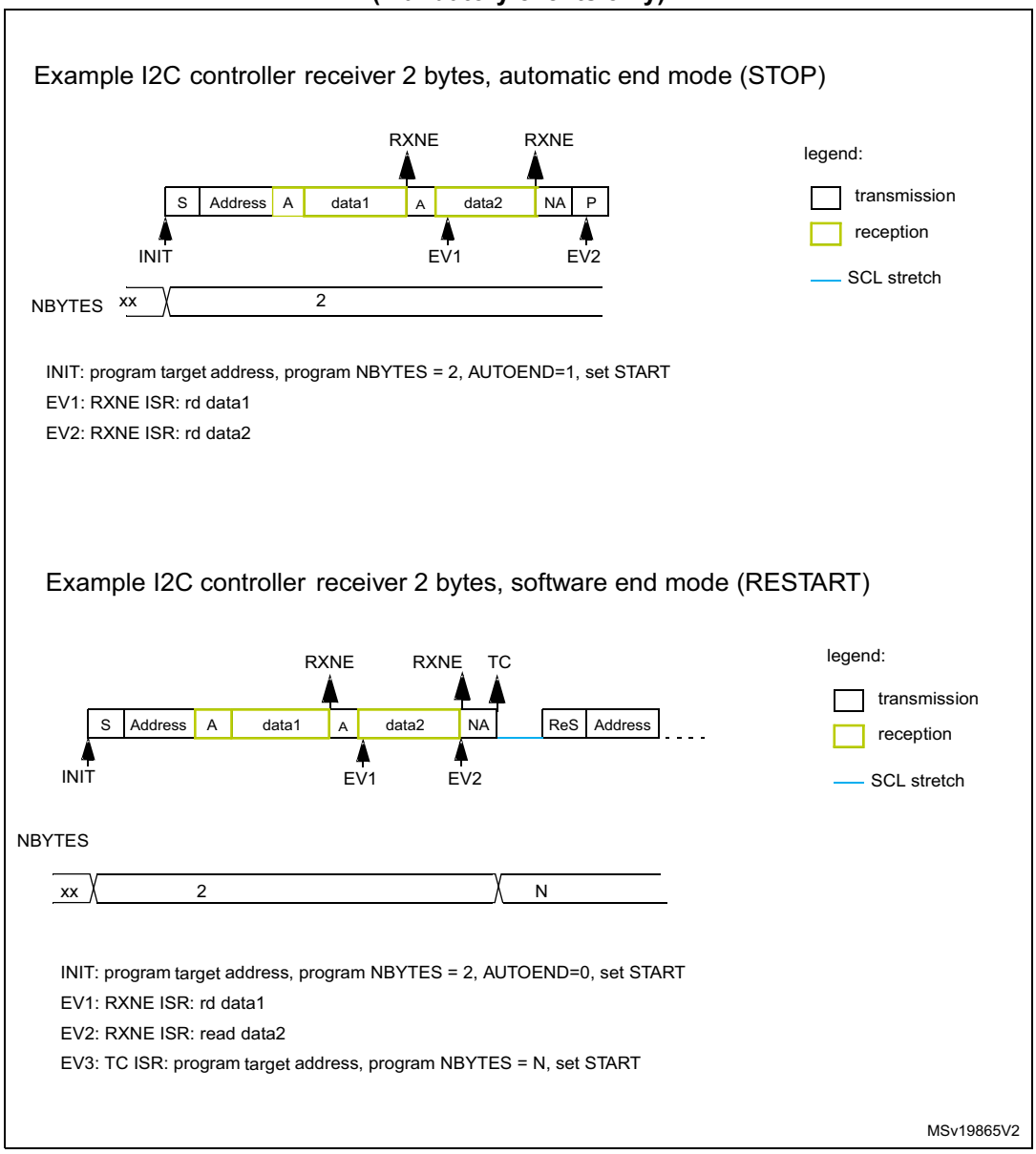


Figure 829. Transfer bus diagrams for I2C controller receiver (mandatory events only)



51.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 588. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 589. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

51.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

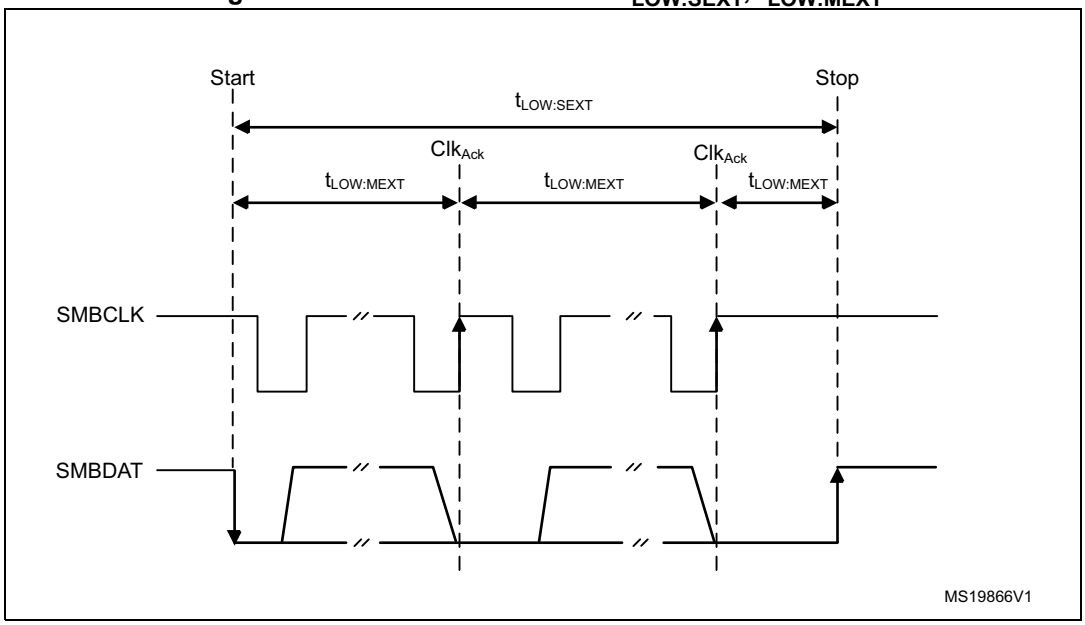
Table 590. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 830. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 51.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

51.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 591. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 592*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 593*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 594*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

51.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 592. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

Table 593. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

Table 594. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs

51.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 51.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 831. Transfer sequence flow for SMBus target transmitter N bytes + PEC

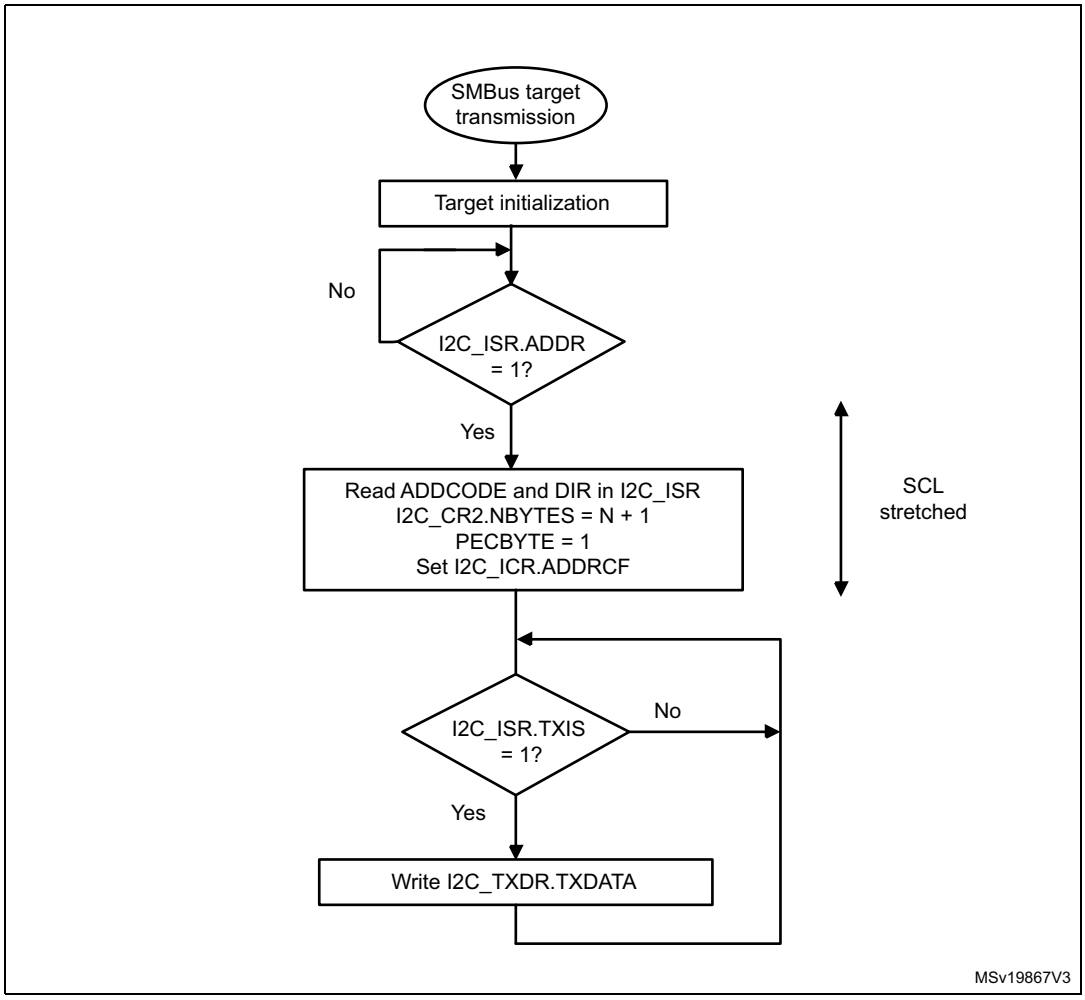
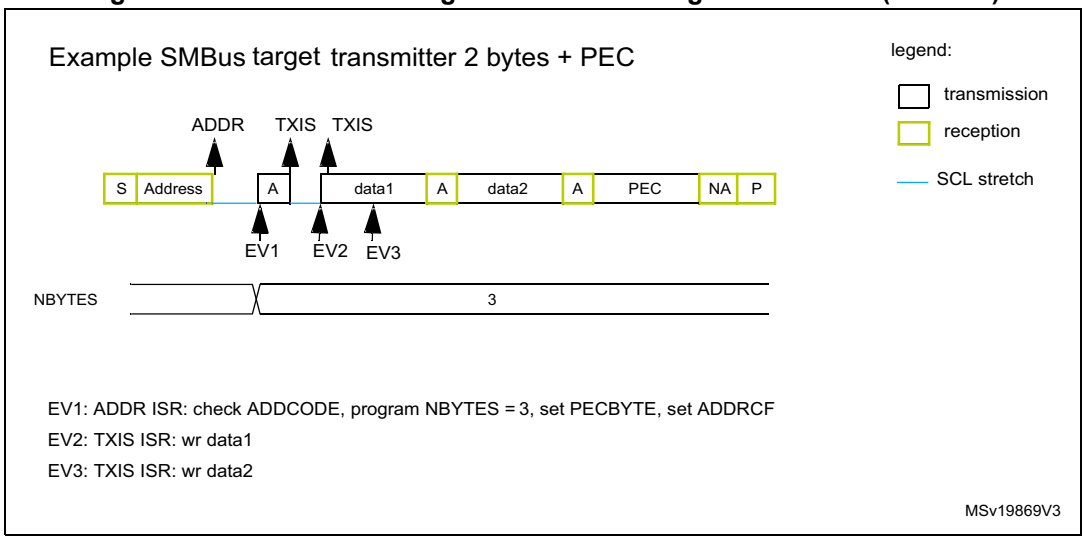


Figure 832. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 833. Transfer sequence flow for SMBus target receiver N bytes + PEC

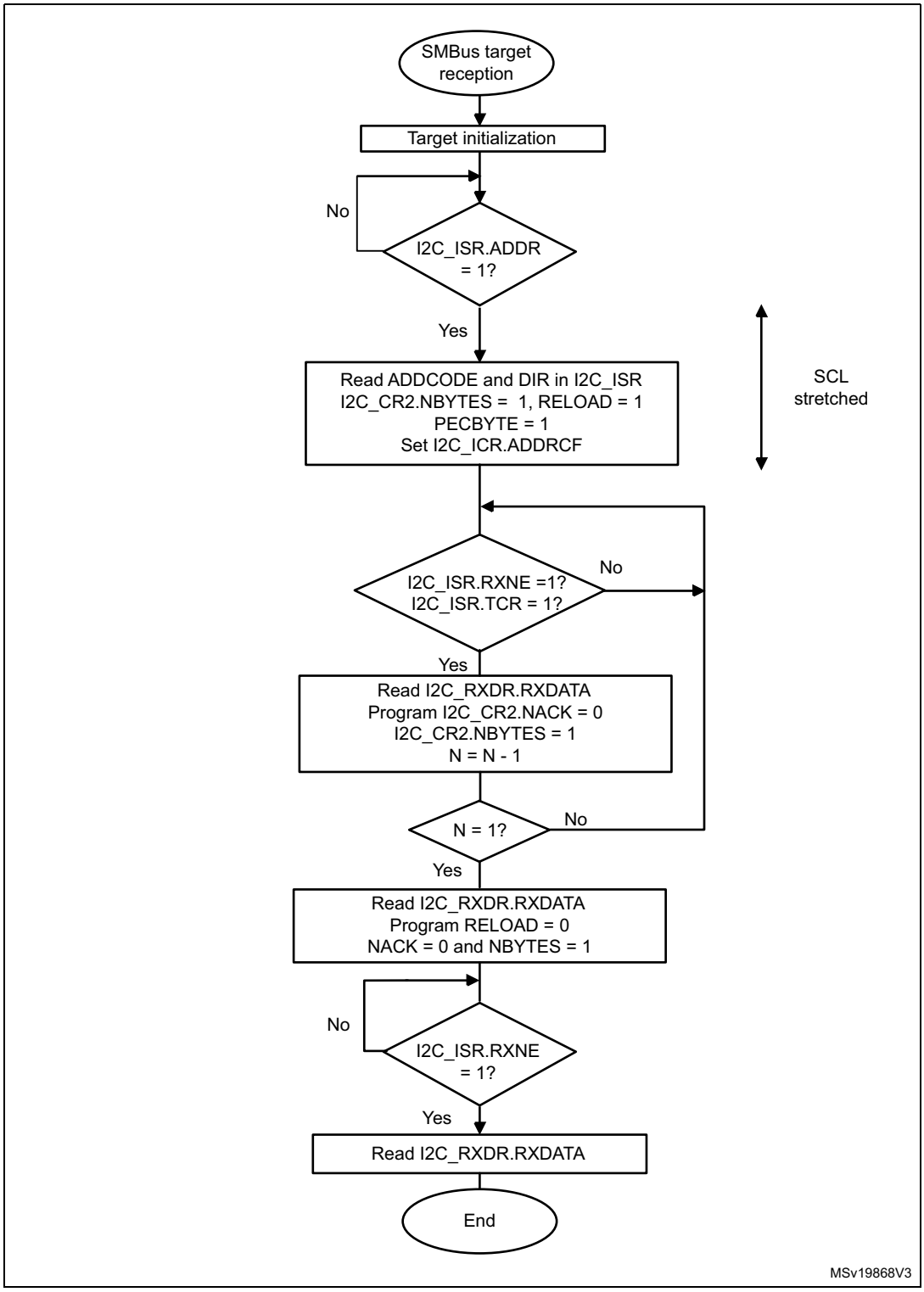
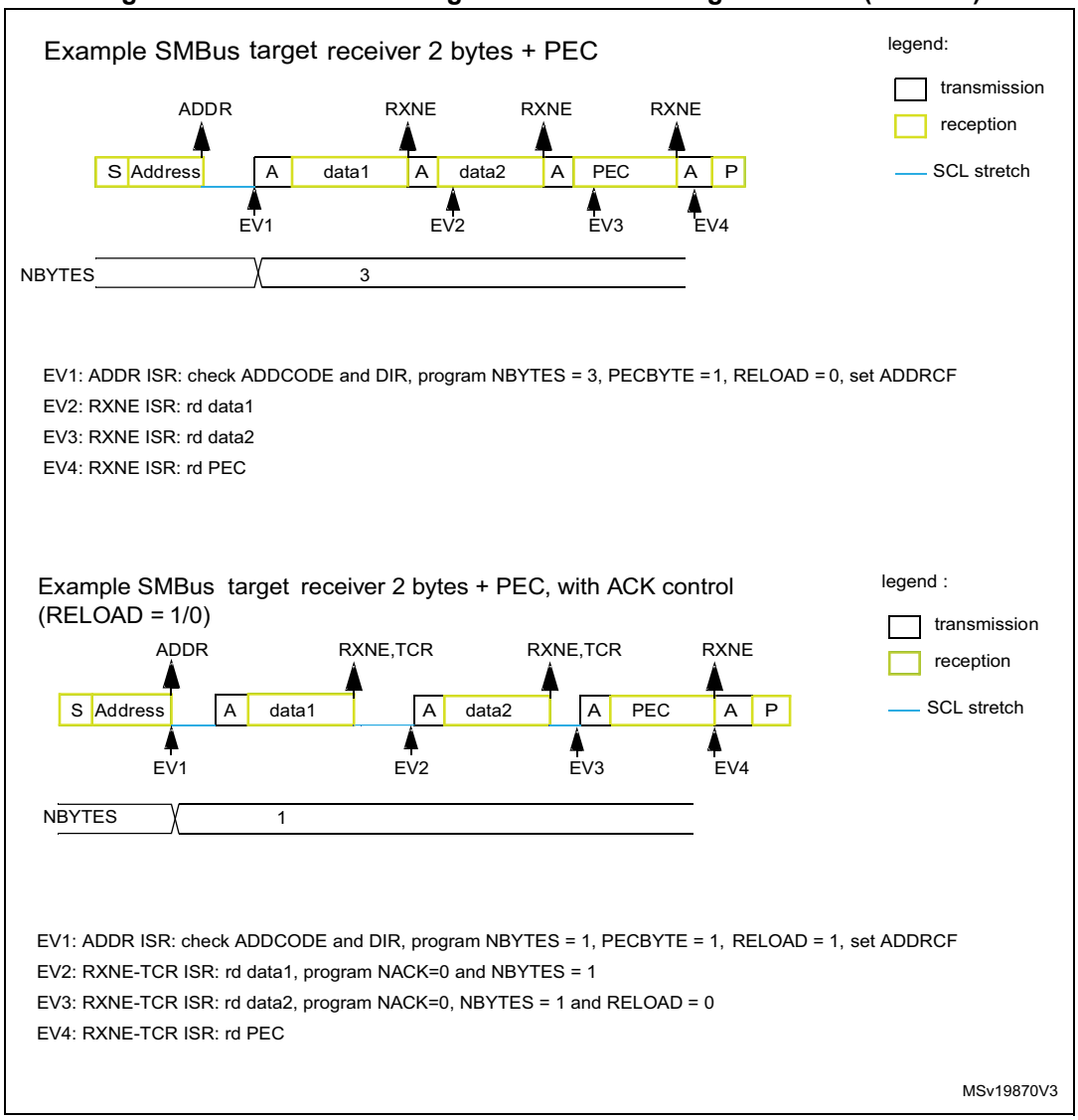


Figure 834. Bus transfer diagrams for SMBus target receiver (SBC = 1)



51.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 51.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

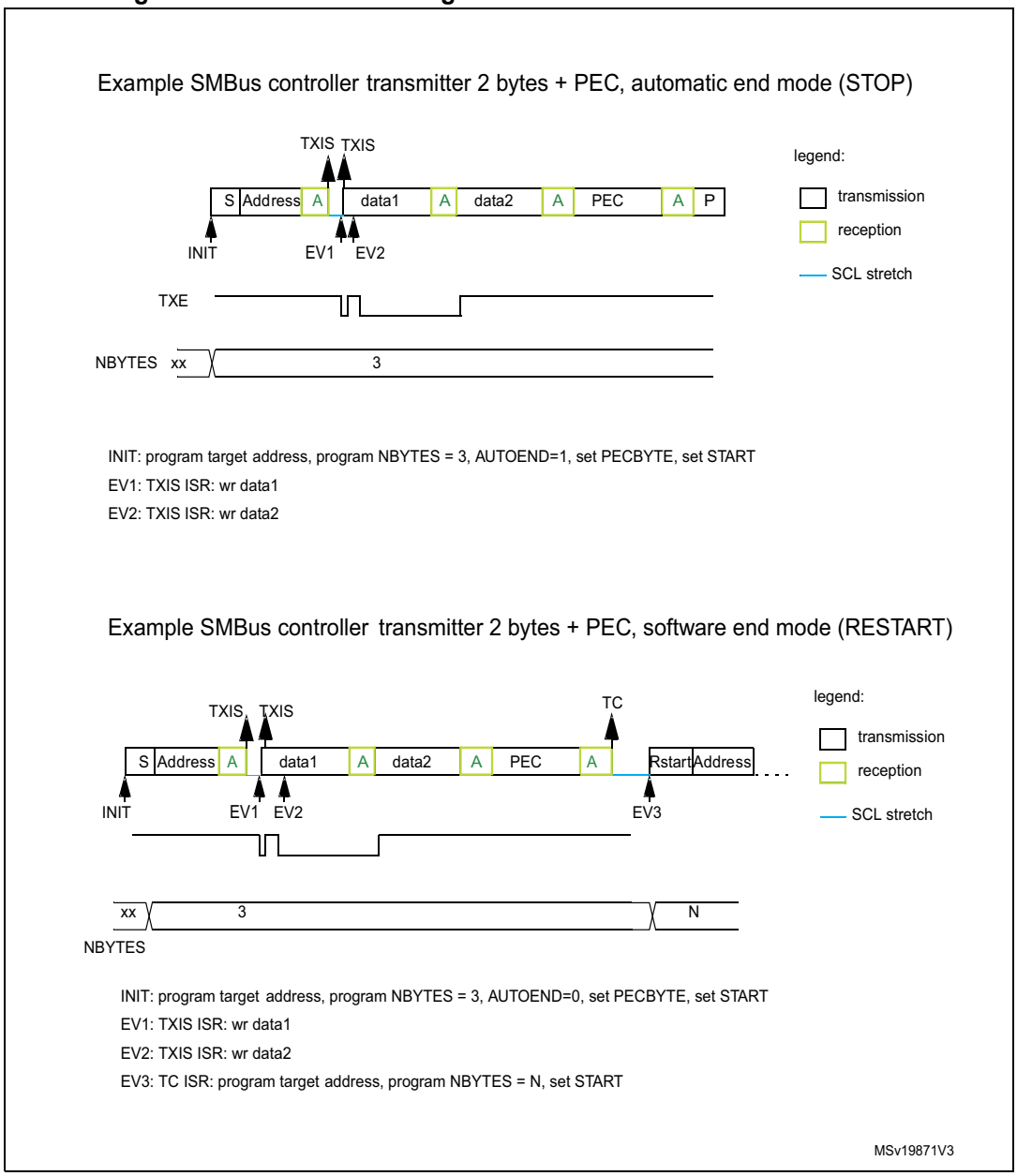
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 835. Bus transfer diagrams for SMBus controller transmitter



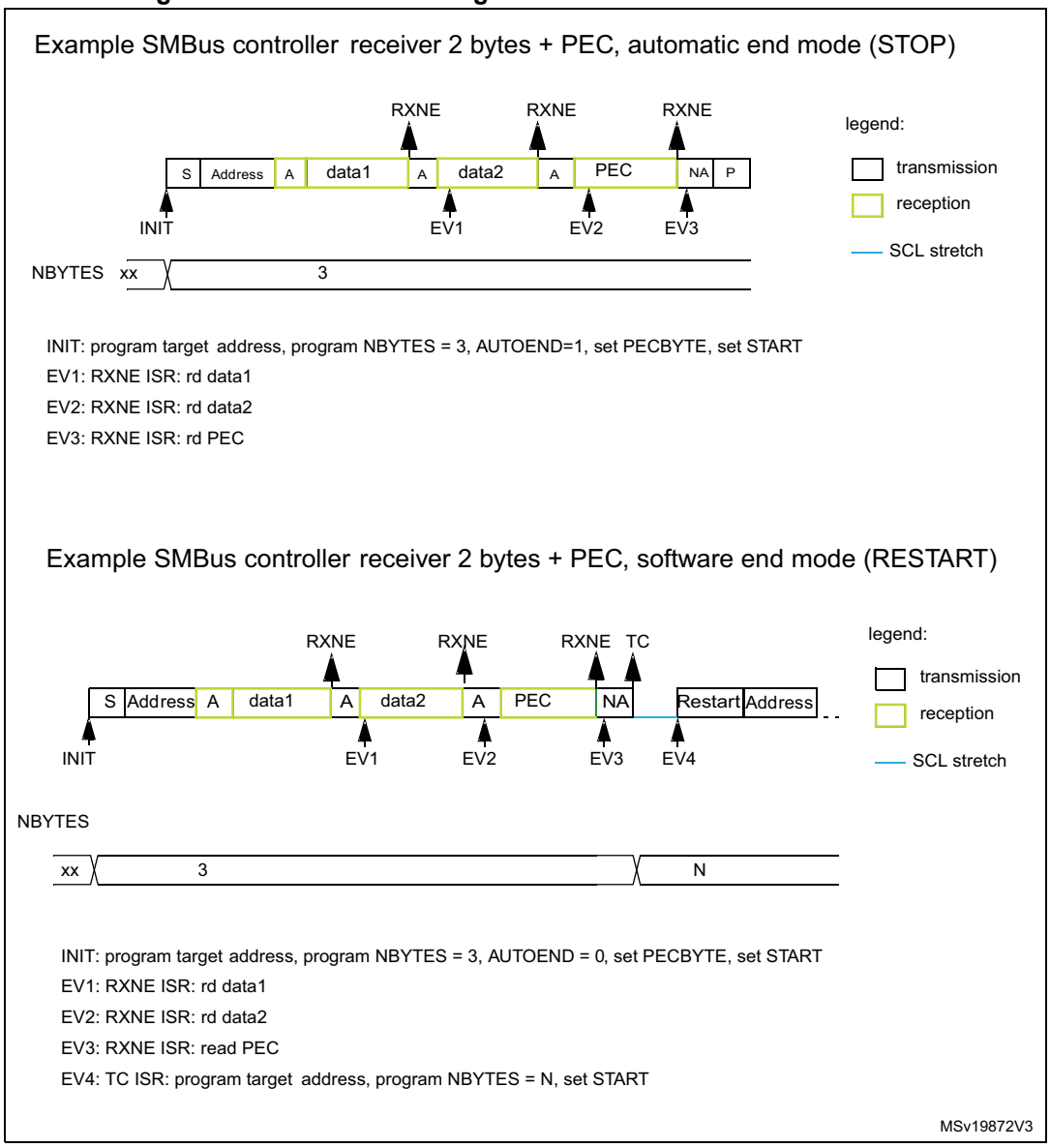
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 836. Bus transfer diagrams for SMBus controller receiver



51.4.16 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI only oscillator must be selected as the clock source for i2c_ker_ck to allow the wake-up from Stop mode.

In Stop mode, the HSI only oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI only oscillator and stretches SCL low until the oscillator wakes up.

HSI only is then used for the address reception.

If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI only oscillator is stopped again and the device does not wake up.

Note: When the system clock is used as I2C clock, or when WUPEN = 0, the HSI only oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a controller or as an addressed target after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: The feature is only available when the HSI only oscillator is selected as the I2C clock.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

51.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

51.5 I2C in low-power modes

Table 595. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI only). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 51.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

51.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 596. I2C interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop modes	Exit Standby modes
I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
	Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
	STOP detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
	Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0			
	Transfer complete	TC		Write START = 1 or STOP = 1			
	Address matched	ADDR	ADDRIE	Write ADDRCF = 1		Yes(1)	
	NACK reception	NACKF	NACKIE	Write NACKCF = 1		No	
I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
	Arbitration loss	ARLO		Write ARLOCF = 1			
	Overrun/underrun	OVR		Write OVRCF = 1			
I2C_ERR	PEC error	PECERR	ERRIE	Write PECERRCF = 1	Yes	No	No
	Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF = 1			
	SMBus alert	ALERT		Write ALERTCF = 1			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to *Section 51.3: I2C implementation*.

51.7 I2C DMA requests

51.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 16: General purpose direct memory access controller (GPDMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

51.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 16: General purpose direct memory access controller (GPDMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

51.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.

51.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

51.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear

0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.

1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear

0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.

1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **FMP**: Fast-mode Plus drive enable

0: FMP I/O drive disabled

1: FMP I/O drive enabled

Bit 23 **PECEN:** PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).

1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).

Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable
0: Device default address disabled. Address 0b1100001x is NACKed.
1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable
0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable
0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.
Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
0: Clock stretching enabled
1: Clock stretching disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control
This bit is used to enable hardware byte control in target mode.
0: Target byte control disabled
1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}
Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generates an interrupt:
- arbitration loss (ARLO)
- bus error detection (BERR)
- overrun/underrun (OVR)
- timeout detection (TIMEOUT)
- PEC error detection (PECERR)
- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable
0: Transfer complete interrupt disabled
1: Transfer complete interrupt enabled
Note: Any of these events generates an interrupt:
Transfer complete (TC)
Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable
0: STOP detection (STOPF) interrupt disabled
1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disabled
1: Peripheral enabled
Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

51.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer

1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.

This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

51.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

51.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable
0: Own address 2 disabled. The received target address OA2 is NACKed.
1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN = 0.
As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

51.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

51.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

– Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

– Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

51.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

51.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

51.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

51.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I²C-bus.

51.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

51.9.12 I2C register map

The table below provides the I2C register map and the reset values.

Table 597. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

52 Improved inter-integrated circuit (I3C)

52.1 I3C introduction

The I3C interface handles communication with other devices such as sensor(s) and host processor.

An I3C bus is a two-wire, serial single-ended, multi-drop bus, intended to improve the legacy I^2C bus. The I3C interface improves the features of the I^2C interface (adding especially in-band interrupts, dynamic address assignment, energy-efficient data rates, power management) meanwhile preserving some backward compatibility: it allows an I^2C target to operate on an I3C bus in legacy I^2C standard-mode (Sm), fast-mode (Fm), or legacy I^2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C SDR-only peripheral implements all the required features and many optional features of the MIPI® I3C specification v1.1 (see *Table 598: I3C peripheral controller/target features versus MIPI v1.1*). It can control all I3C bus-specific sequencing, protocol, arbitration, and timing.

After reset, the software can initialize the I3C peripheral as primary controller or as target:

- As I3C controller, the software can enable the I3C peripheral and initialize the I3C bus.
- As I3C target, the software can enable the I3C peripheral to operate on an I3C bus.

The I3C peripheral can be used with DMA, to off-load the CPU.

52.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in *Table 598*), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I^2C read/write messages to legacy I^2C targets in Sm/Fm/Fm+
  - I3C SDR read/write private messages

  - I3C SDR broadcast CCC messages (see details in *Table 608*)
  - I3C SDR read/write direct CCC messages (see details in *Table 608*)
- Frame-level management, when controller:
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four I3C targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with x = 0, 1, 2, 3 when controller
  - TEx with x = 0, 1, ... , 6 when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see *Section 52.3.2*):
  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see *Section 52.3.2*):
  - On a reset pattern
  - On a missed start on the serial bus
- Multi-clock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in *Section 52.6.2*)

52.2.1 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in *Table 598*.

Table 598. I3C peripheral controller/target features versus MIPI v1.1

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		When controller	When target	
I3C SDR message	X	X	X	-
Legacy I^2C message (Sm/Fm/Fm+)	X	X	-	Mandatory when controller if the I3C bus is mixed with (external) legacy I^2C target(s). Optional in MIPI v1.1 when target.
HDR DDR message	X	-	-	Optional in MIPI v1.1
HDR-TSL/TSP, HDR-BT	X	-	-	Optional in MIPI v1.1
Dynamic address assignment	X	X	X	Supported broadcast ENTDAA when controller and target. Supported broadcast SETAASA and direct SETDASA when controller.
7-bit static I^2C address	X	X	-	No support of the I3C peripheral as target on an I^2C bus.
Grouped addressing	X	X	-	Optional in MIPI v1.1
CCCs	X	X	X	Mandatory and some optional CCCs supported (refer to *Table 608* when controller/target).
Error detection and recovery	X	X	X	-
In-band interrupt (with MDB)	X	X	X	-
Secondary controller	X	X	X	-
Hot-join mechanism	X	X	X	-
Target reset	X	X	X	-
Synchronous timing control	X	X	-	Optional in MIPI v1.1
Asynchronous timing control 0	X	X	-	Mandatory in MIPI v1.1 when controller (refer to *Timing control and sensor acquisition timestamping when I3C controller*). Optional in MIPI v1.1 when target.
Asynchronous timing control 1, 2, 3	X	-	-	Optional in MIPI v1.1
Device to device tunneling	X	X	-	Optional in MIPI v1.1
Multi-lane data transfer	X	-	-	Optional in MIPI v1.1
Monitoring device early termination	X	-	-	Optional in MIPI v1.1

52.2.2 I3C without clock stretching by the target

The I3C bus is intended to deprecate the I²C bus. The I²C specification optionally features clock stretching by the target, but this is not allowed by the MIPI® I3C specification.

The user may have to port this I²C feature when the I²C bus is replaced by an I3C bus. This section details four main alternative methods to replace clock stretching by the target for the same applicative purpose.

The I²C SCL is open drain. (Optional) I2C clock stretching by the target pauses a transaction by holding the SCL line low:

- It is generally used at byte level, when the target software needs more time to store a received byte or to prepare another byte for transmission.
- It may be used at bit level, typically by an MCU without an I2C bus for software emulation of I²C SCL and SDA lines over GPIOs.

The main advantage of this feature is on the target side; it is convenient for the application software of the target device to have direct control on the I²C bus, and to suspend an I²C transaction when it needs more time before answering.

This feature has a disadvantage on the bus and controller side since there is no suspension timeout in the I²C specification. The I²C bus and controller can become permanently hanged by a lost target. There is no in-band timeout. Bus error recovery is not possible without another alert signal (for example SMBus).

This I²C target capability is not allowed by the MIPI® I3C specification when a target is operating on theI3C bus: the I3C SCL line is driven in push-pull mode by the controller.

Note: The I3C SDA line is a mix of push-pull and open-drain phases.

MIPI® I3C relies on the following four alternative methods to guarantee that a target always has enough time to respond to an I3C or legacy I²C message:

- *I3C byte-level a read data ACK/NACK*
- *I3C/I2C target needs a clock stalling of up to 100 µs (at different phases)*
- *IBI with pending read: for read, if I3C target needs a clock stalling above 100 µs*
- *Maximum read turnaround time: for read, if I3C target needs a clock stalling above 100 µs*

I3C byte-level a read data ACK/NACK

An I3C target can ACK/NACK at byte level a read data

- Any controller or addressed target (new feature in I3C) can terminate the number of exchanged data at a byte level.
- A target needing more time can terminate the transfer early by driving SDA low on the 9th T bit.
- The controller can again ask for new bytes to be read if more bytes are expected from the target.

This is the contrary of I²C where only the controller can terminate a read transfer.

- The I²C controller cannot distinguish whether or not a data byte was transmitted by a (slow) target.
- To prevent this I²C protocol issue, the I²C target must be able to stretch the clock.

I3C/I^{2}C target needs a clock stalling of up to 100 µs (at different phases)

If the delay requested by the target is less than 100 µs, MIPI® has standardized optional features for the I3C controller to stall-low the SCL clock up to 100 µs.

This I3C peripheral, when acting as controller, fully supports clock stalling and can be reconfigured without disabling the I3C controller, with:

- One programmable value of up to 100 µs, with a typical granularity of 1µs.
- Seven independent programmable enable bits, to possibly stall at the following I3C bus phases:
  - On I^{2}C legacy address ACK/NACK bit
  - On I^{2}C legacy write byte-level data ACK/NACK bit
  - On I^{2}C legacy read byte-level data byte ACK/NACK bit
  - On I3C address ACK/NACK bit
  - On I3C CCC command parity bit
  - On I3C write byte-level data parity bit
  - On I3C read byte-level data parity bit

Note: This I3C peripheral as controller allows more than 100 µs; the programmed delay is relative to the kernel clock frequency.

IBI with pending read: for read, if I3C target needs a clock stalling above 100 µs

- The I3C target can initiate an IBI with a pending read notification (optional MIPI feature) when it is ready to be read.
- This I3C peripheral, both as controller and target, supports IBI with pending read notification.
- The controller is aware of the IBI MDB support for pending read notification from a target, by sending the direct GETCAPS CCC format 1 (returned bit 6 of the GETCAP3 byte).
- The I3C target issues an IBI with MDB[7:5] = 3'b101 without any optional payload, to notify to the controller that it is ready to be read.
- The I3C controller must then issue a private read on the next transaction to this target.

Maximum read turnaround time: for read, if I3C target needs a clock stalling above 100 µs

- If the delay requested by the target is more than 100 µs, the controller can get the maximum read turnaround time by a direct GETMXDS CCC (format 2, 5 bytes).
- The maximum read turnaround time is returned as 24 bits, for a value between 0 and 16 seconds.
- The controller then inserts a STOP between a write (index) and a read.

52.3 I3C implementation

52.3.1 I3C instantiation

There is a single I3C instance in the device.

52.3.2 I3C wake-up from low-power mode(s)

The I3C peripheral can wake up the device from a low-power mode, as detailed in *Table 600*. For more details about the wake-up capabilities, refer to *Section 52.13*.

Table 600. I3C wake-up

Wake-up
From Stop mode

52.3.3 I3C FIFOs

The FIFOs are implemented as defined in *Table 601*.

Table 601. I3C FIFOs implementation

FIFO	Content	Unit	Size (in unit)	Used as controller/target (rationale)
C-FIFO	(32-bit) Control words	Word	2	Controller (a frame can be based on multiple control words; this is not the case as target)
S-FIFO	(32-bit) Status words			Controller (target: status only in register mode)
TX-FIFO	Transmitted data	Byte	8	Controller and target
RX-FIFO	Received data			

52.3.4 I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

52.3.5 I3C interrupt(s)

The interrupt mapping is implemented as detailed in *Table 602*.

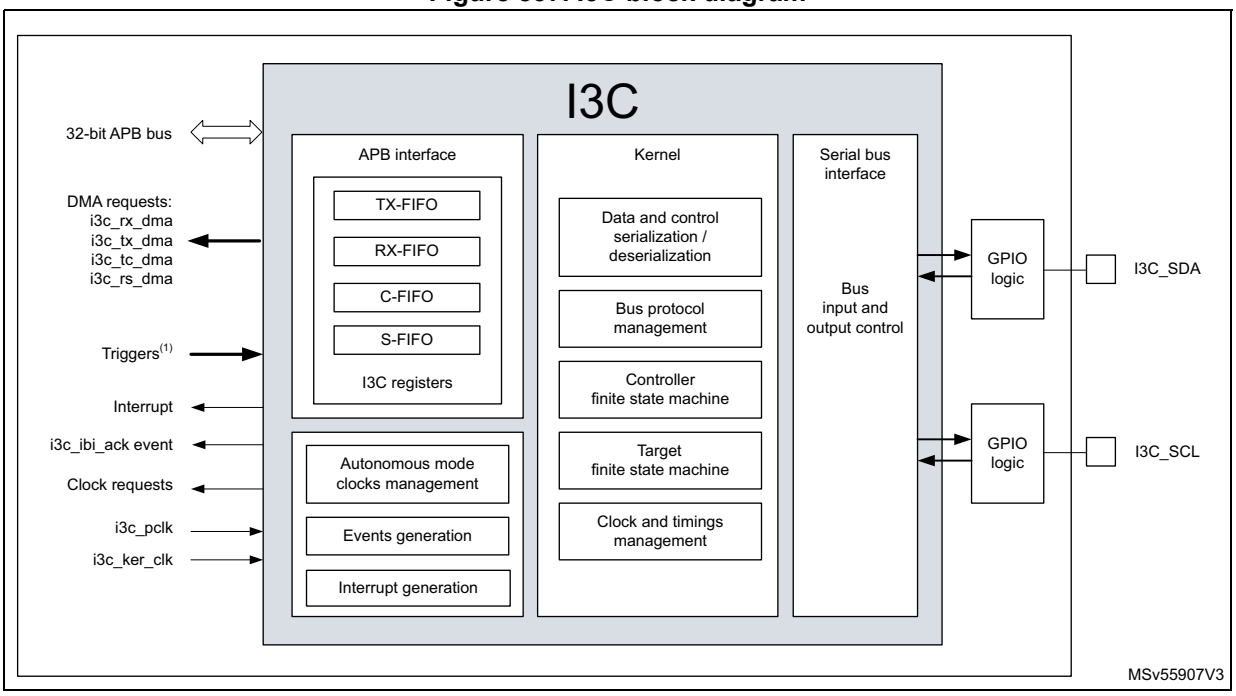
Table 602. I3C interrupt(s)

Signal name	Signal type	Description
i3c_err_it	O	Error interrupt line
i3c_evt_it	O	Event interrupt line

52.4 I3C block diagram

The I3C block diagram is illustrated in *Figure 837*.

Figure 837. I3C block diagram



1. Refer to *Section 52.3.4: I3C triggers*.

52.5 I3C pins and internal signals

Table 603. I3C input/output pins

Pin name	Pin type	Description
I3C_SDA	Input/output	Serial bus data line
I3C_SCL	Input/output	Serial bus clock line

Note: During initialization, the selected GPIOs for I3C_SCL and I3C_SDA should be configured as alternate function and no pull.

Table 604. I3C internal input/output signals

Signal name	Signal type	Description
i3c_pclk	I	APB clock
i3c_ker_clk	I	Kernel clock (also named as I3CCLK)
i3c_pclk_req	O	APB clock request
i3c_ker_clk_req	O	Kernel clock request
i3c_it(1)	O	Global interrupt line

Table 604. I3C internal input/output signals (continued)

Signal name	Signal type	Description
i3c_err_it(1)	O	Error interrupt line
i3c_evt_it(1)	O	Event interrupt line
i3c_rx_dma	O	DMA request for reading received bytes/words from RX-FIFO
i3c_tx_dma	O	DMA request for writing to be transmitted bytes/words to TX-FIFO
i3c_tc_dma	O	DMA request for writing to be transmitted control words to C-FIFO, when the I3C peripheral acts as controller
i3c_rs_dma	O	DMA request for reading status words from S-FIFO, when the I3C peripheral acts as controller
i3c_ibi_ack	O	IBI acknowledge event when the I3C peripheral acts as controller, for timing control with asynchronous mode 0

1. Refer to *Section 52.3.5: I3C interrupt(s)*.

52.6 I3C reset and clocks

52.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the I3C peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled I3C peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to *Table 608* and *Section 52.16.17* for more details.

This reset interrupt notification can be used to wake up from a low power mode.

For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

52.6.2 I3C clocks and requirements

As indicated in *Figure 837*, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming *I3C timing register 0 (I3C_TIMINGR0)*, *I3C timing register 1 (I3C_TIMINGR1)* and *I3C timing register 2 (I3C_TIMINGR2)*, as summarized in *Controller initialization* and *Updating the configuration for a transfer, as controller*.
  - When target: the user must set and comply with the bus available condition (t_{AVAL} for an in-band interrupt or controller-role request), and the bus idle condition (t_{IDLE} for a hot-join request), by programming *I3C timing register 1 (I3C_TIMINGR1)*, as summarized in *Target initialization*.

- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to section Reset and clock control (RCC) for more details).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

Note: Sustaining $F_{SCL\ max}$ = 12.9 MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < t_{HIGH} (SCL clock high period)
   - $t_{HIGH\ min}$ = 24 ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < t_{CASr} (clock after repeated start condition)
   - $t_{CASr\ min}$ = $t_{CAS\ min}$ / 2 = 19.2 ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < t_{LOW_OD} (SCL clock low period in open drain)
   - $t_{LOW_OD\ min}$ = 200 ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL\ max}$ = 12.9 MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3x\ F_{I3CCLK} - F_{SCL})]$

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.

52.7 I3C peripheral state and programming

52.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see *Figure 838* and *Figure 839*), the peripheral is in one of the following states:

- Disabled state:
  - After an I3C reset (system reset or I3C reset from RCC), the I3C peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the I3C peripheral takes into account the value of the different initialized configuration registers, and switches to the (enabled and) idle state. The peripheral is initialized as I3C target or I3C controller.
  - As I3C controller, then software can enable the I3C peripheral and initialize the I3C bus.
  - As I3C target, then the software can enable the I3C peripheral to operate on an I3C bus.
- Idle state:
  - After being enabled (EN = 1), the I3C peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see *Updating the configuration for a transfer, as controller* and *Updating the configuration of the I3C peripheral, as target* for more details.
  - The I3C peripheral switches to the (enabled and) active state in either one of the following conditions:

    Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request).

    Once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write).
- Active state:
  - The I3C peripheral executes the transfer(s) on the bus.
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the *I3C event register (I3C_EVR)*, and the corresponding interrupt is enabled by *I3C interrupt enable register (I3C_IER)*. The I3C peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured:

    As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF).

    As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF).

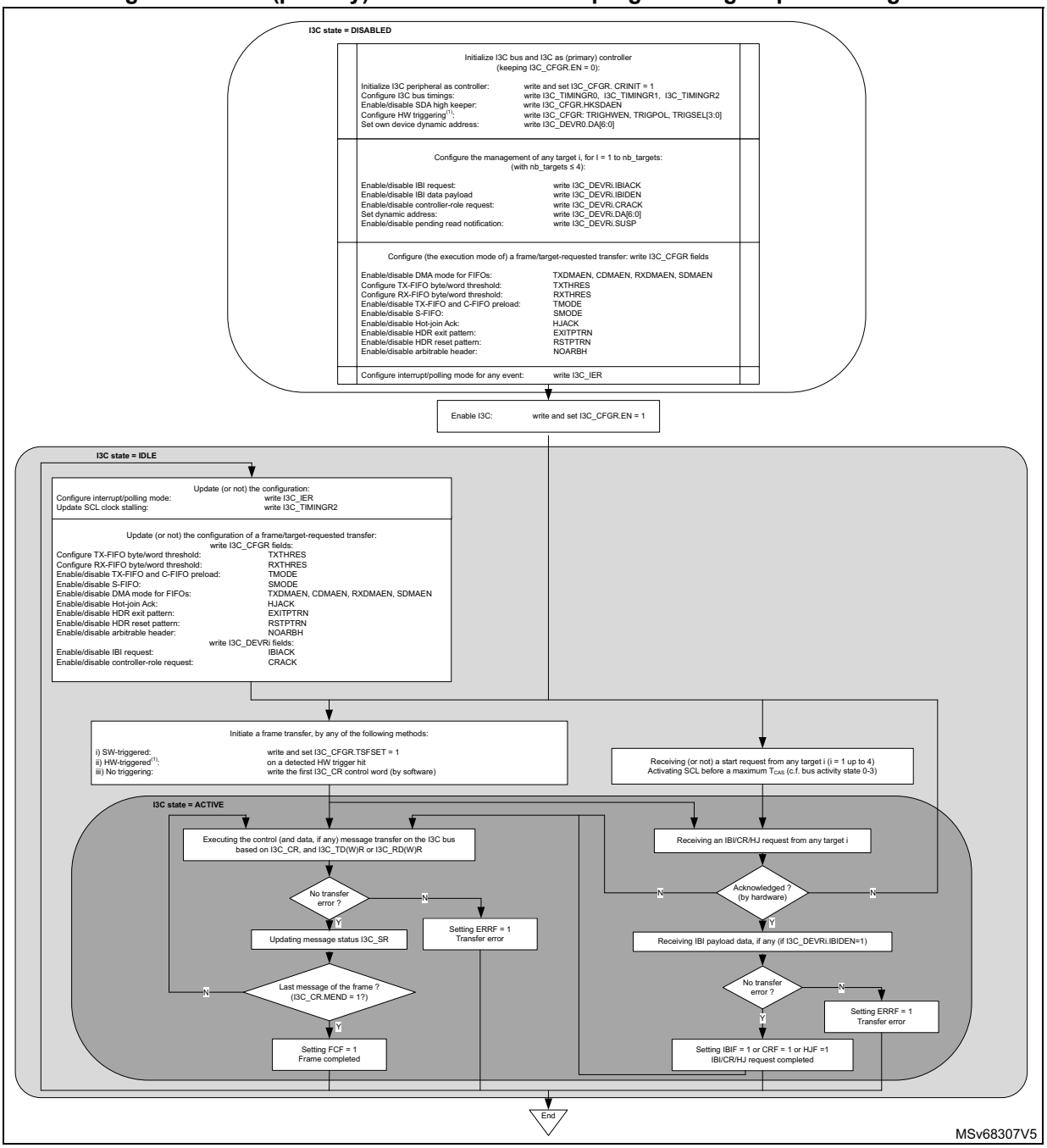
Note: The software can disable the I3C peripheral (write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.

Note: The software can disable the generation of the FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 52.3), to stream a series of frames.

52.7.2 I3C controller state and programming sequence

Figure 838 illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 838. I3C (primary) controller state and programming sequence diagram



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 52.3.4: I3C triggers*.

Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
- Configure I3C bus timings:
  a) *I3C timing register 0 (I3C_TIMINGR0)*:
  - SCL clock high time period (t_{DIG_H}, $t_{DIG_H_MIXED}$) in legacy I^2C and I3C open-drain/push-pull
  - SCL clock low time period (t_{DIG_L}, $t_{DIG_OD_L}$) in legacy I^2C and I3C open-drain/push-pull

  b) *I3C timing register 1 (I3C_TIMINGR1)*:
  - SDA hold time in push-pull (t_{HD_PP})
  - bus free condition time (I3C $t_{CAS,}$ legacy I^2C t_{BUF})
  - I3C repeated start timing (t_{CASr}, t_{CBSr})
  - I3C stop timing (t_{CBP})
  - SCL clock low maximum stalling on the ENTDAA CCC ($t_{STALLDAA}$), or on the ACK/NACK of a legacy I^2C or address phase of an I3C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I^2C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I^2C write (t_{STALL}), to adjust SCL clock low stalling, if needed by the I3C peripheral itself, when used as controller
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)

  c) *I3C timing register 2 (I3C_TIMINGR2)*:
  - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I^2C target
- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*
- Configure the management of any device target x: *I3C device x characteristics register (I3C_DEVRx)*, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
  - FCFDIS: FCF generation enable/disable
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

- Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
  - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
- No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

- If the C-FIFO in not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)
- If the C-FIFO in configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum t_{CAS} time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in *Section 52.9*:

- *Figure 850: IBI transfer, as controller/target*
- *Figure 851: Hot-join request transfer, as controller/target*
- *Figure 852: Controller-role request transfer, as controller/target*

Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.

For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in *Section 52.9*:

- *Figure 840: I3C CCC messages, as controller*
- *Figure 841: I3C broadcast ENTDAA CCC, as controller*
- *Figure 842: I3C broadcast, direct read and direct write RSTACT CCC, as controller*
- *Figure 847: I3C private read/write messages, as controller*
- *Figure 849: Legacy I2C read/write messages, as controller*

Figure 838 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in *Section 52.10*.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating *I3C status register (I3C_SR)*, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to *Section 52.10.4*.

Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via *I3C timing register 2 (I3C_TIMINGR2)*
- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify the following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH
  - FCFDIS

- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - *I3C message control register (I3C_CR)*, *I3C message control register [alternate] (I3C_CR)*
  - *I3C status register (I3C_SR)*
  - *I3C transmit data byte register (I3C_TDR)*, *I3C transmit data word register (I3C_TDWR)*
  - *I3C receive data byte register (I3C_RDR)*, *I3C receive data word register (I3C_RDWR)*
- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via *I3C device x characteristics register (I3C_DEVRx)*

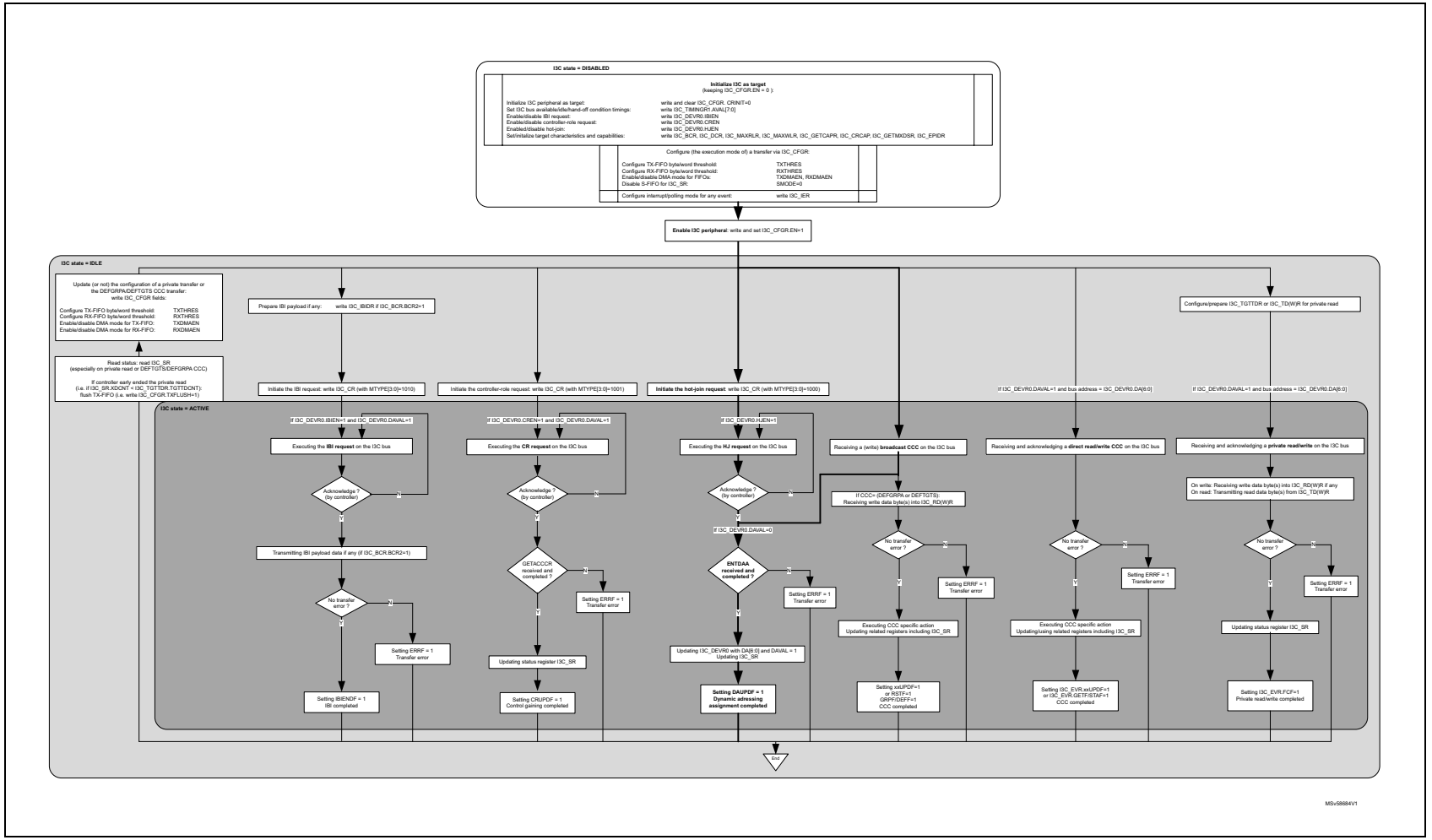
The registers usage versus the I3C peripheral role as controller is summarized in *Section 52.8.1*.

The static/dynamic registers fields usage when acting as controller is summarized in *Table 606*.

52.7.3 I3C target state and programming sequence

Figure 839 illustrates the overall programming sequence of the I3C peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 839. I3C target state and programing sequence diagram



Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set the I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
- Set the I3C bus timings via *I3C timing register 1 (I3C_TIMINGR1)*: write AVAL[7:0] for:
  - Bus available condition time (t_{AVAL}) for IBI or controller-role request
  - Bus idle condition time (t_{IDLE}) for hot-join request
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write *I3C own device characteristics register (I3C_DEVR0)* with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - *I3C bus characteristics register (I3C_BCR)*
  - *I3C device characteristics register (I3C_DCR)*
  - *I3C maximum read length register (I3C_MAXRLR)*
  - *I3C maximum write length register (I3C_MAXWLR)*
  - *I3C get capability register (I3C_GETCAPR)*
  - *I3C controller-role capability register (I3C_CRCAPR)*
  - *I3C get max data speed register (I3C_GETMXDSR)*
  - *I3C extended provisioned ID register (I3C_EPIDR)*
- Configure the execution mode of a transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message on the I3C bus

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.

After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 52.9*:

- *Figure 843: I3C CCC messages, as target*
- *Figure 844: I3C broadcast ENTDAA CCC, as target*
- *Figure 845: I3C broadcast DEFTGTS CCC, as target*
- *Figure 846: I3C broadcast DEFGRPA CCC, as target*
- *Figure 848: I3C private read/write messages, as target*

Figure 839 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 52.10*.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the bus by updating the *I3C status register (I3C_SR)*, which can be read by the software after being notified by the corresponding flag in the *I3C event register (I3C_EVR)*, or by the corresponding interrupt if enabled in the *I3C interrupt enable register (I3C_IER)*.

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer on the I3C bus

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in *Section 52.9*:

- *Figure 850: IBI transfer, as controller/target*
- *Figure 851: Hot-join request transfer, as controller/target*
- *Figure 852: Controller-role request transfer, as controller/target*

Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the *I3C IBI payload data register (I3C_IBIDR)*, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write *I3C message control register [alternate] (I3C_CR)* with MTYPE[3:0] = 1010)
- Modify/prepare *I3C target transmit configuration register (I3C_TGTTDR)*, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC)

The registers usage vs. the I3C peripheral role as target is summarized in *Section 52.8.1*.

The static/dynamic registers fields usage, when acting as target, is summarized in *Table 607*.

52.8 I3C registers and programming

52.8.1 I3C register set, as controller/target

Table 605 lists the registers and their usage versus the I3C peripheral role.

Table 605. I3C register usage

Register	Used as controller	Used as target
I3C_CR	X	X
I3C_CFGR	X	X
I3C_RDR	X	X
I3C_RDWR	X	X
I3C_TDR	X	X
I3C_TDWR	X	X
I3C_IBIDR	X	X
I3C_TGTTDR	-	X
I3C_SR	X	X
I3C_SER	X	X
I3C_RMR	X	X
I3C_EVR	X	X
I3C_IER	X	X
I3C_CEVR	X	X
I3C_MISR	X	X

Table 605. I3C register usage (continued)

Register	Used as controller	Used as target
I3C_DEVR0	X	X
I3C_DEVRx (x = 1 to 4)	X	-
I3C_MAXRLR	-	X
I3C_MAXWLR	-	X
I3C_TIMINGR0	X	-
I3C_TIMINGR1	X	X
I3C_TIMINGR2	X	-
I3C_BCR	-	X
I3C_DCR	-	X
I3C_GETCAPR	-	X
I3C_CRCAPR	-	X
I3C_GETMXDSR	-	X
I3C_EPIDR	-	X

52.8.2 I3C registers and fields use versus peripheral state, as controller

When the I3C peripheral acts as controller, *Table 606* lists the registers and their usage versus the controller state (disabled, idle, and active).

Table 606. I3C registers/fields usage versus controller state

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	-	X

Table 606. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CFGR	X	CRINIT HKSDAEN EN(1)	Write: any used field except {CRINIT, HKSDAEN}, namely {CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH}(2) FCFDIS CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	-	Read
I3C_TGTTDR	-			
I3C_SR	X	-	-	Read
I3C_SER	X	-	Read	-
I3C_RMR	X	-	-	Read RADD[6:0] IBIRDCNT[2:0]

Table 606. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (controller-role fields): HJF CRF IBIF FCF ERRF RXTGTENDF RXFNEF TXFNFF SFNEF CFNFF RXLASTF TXLASTF TXFEF CFEF
I3C_IER	X	Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)		
I3C_CEVR	X	-	-	Write (controller-role fields, refer to I3C_EVR)
I3C_MISR	X	-	-	Read (controller-role fields): HJMIS CRMIS IBIMIS FCMIS ERRMIS RXTGTENDMIS RXFNEMIS TXFNFMIS SFNEMIS CFNFMIS
I3C_DEVR0	X	Write DA[6:0](2)		-
I3C_DEVRx x = 1..4	X	Write SUSP, IBIDEN, IBIACK, CRACK, DA[6:0](2)		-
I3C_MAXRLR	-			
I3C_MAXWLR	-			
I3C_TIMINGR0	X	X	-	-
I3C_TIMINGR1	X	X	-	-
I3C_TIMINGR2	X	Write(2)		
I3C_BCR	-			
I3C_DCR	-			
I3C_GETCAPR	-			
I3C_CRCAPR	-			

Table 606. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_GETMXDSR	-			
I3C_EPIDR	-			

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).

52.8.3 I3C registers and fields usage versus peripheral state, as target

When the I3C peripheral acts as target, *Table 607* lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 607. I3C registers/fields usage versus target state

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	MTYPE[3:0] DCNT[2:0]	-
I3C_CFGR	X	CRINIT EN(1)	Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES}(2) TXFLUSH RXFLUSH	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	Write	-
I3C_TGTTDR	X	-	Write	-
I3C_SR	X	-	-	Read DIR, XDCNT[15:0]
I3C_SER	X	-	Read DOVR, STALL, PERR, CODERR[3:0]	-
I3C_RMR	X	-	-	Read RCODE[7:0]

Table 607. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (target-role fields): GRPF DEFF INTUPDF ASUPDF RSTF MRLUPDF MWLUPDF DAUPDF STAF GETF WKPF CRUPDF IBIENDF ERRF FCF RXFNEF TXFNFF TXLASTF TXFEF
I3C_IER	X	Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)		
I3C_CEVR	X	-	-	Write (target-role fields, refer to I3C_EVR)
I3C_MISR	X			Read (target-role fields): GRPMIS DEFMIS INTUPDMIS ASUPDMIS RSTMIS MRLUPDMIS MWLUPDMIS DAUPDMIS STAMIS GETMIS WKPMIS CRUPDMIS IBIENDMIS ERRMIS FCMIS RXFNEMIS TXFNFMIS

Table 607. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_DEVR0	X	HJEN CREN IBIEN	Read RSTVAL, RSTACT[1:0], and AS[1:0]	-
I3C_DEVRx x = 1..4	-			
I3C_MAXRLR	X	X	-	-
I3C_MAXWLR	X	X	-	-
I3C_TIMINGR0	-			
I3C_TIMINGR1	X	AVAL[7:0]	-	-
I3C_TIMINGR2	-			
I3C_BCR	X	X	-	-
I3C_DCR	X	X	-	-
I3C_GETCAPR	X	X	-	-
I3C_CRCAPR	X	X	-	-
I3C_GETMXDSR	X	X	-	-
I3C_EPIDR	X	X	-	-

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.

52.9 I3C bus transfers and programming

52.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the I3C peripheral acting as controller or target, is specified in *Table 608*.

Table 608. List of supported I3C CCCs, as controller/target

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Broadcast CCCs							
ENEC	0x00	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x01			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x02 ... 0x05			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
RSTDAA	0x06			-	X	X, DAUPDF	Clear I3C_DEVR0.DAVAL = 0
ENTDAA	0x07			-	X	X, DAUPDF if DAVAL = 0	Update I3C_DEVR0: DA[6:0] and set DAVAL = 1
DEFTGTS	0x08			With [1+ 4x (1+ number_of_targets)] x data bytes	X	X, DEFF	Update I3C_RDR/ I3C_RDWR. Refer to *Figure 845*.
SETMWL	0x09			With two data byte	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x0A			With 2 or 3 data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
ENTTM	0x0B			With one data byte	X	-	-
SETXTIME	0x28		With sub-command byte	Without or with one or more data bytes	X	-	-
SETAASA	0x29		No defining/sub-command byte	No data byte	X	-	-
RSTACT	0x2A		With defining byte (0x00, 0x01 or 0x02)		X	X, RSTF after detected reset pattern	Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
DEFGRPA	0x2B		No defining/sub-command byte	With several data bytes	X	X, GRPF	Update I3C_RDR/ RDWR. Refer to *Figure 846*.
RSTGRPA	0x2C			No data byte	X	-	-

Table 608. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Direct CCCs							Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)
ENEC	0x80	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x81			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x82...0x85			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
SETDASA	0x87			No data byte	X	-	-
SETNEWDA	0x88			With one data byte	X	X, DAUPDF if DAVAL = 1	Update I3C_DEVR0: DA[6:0] (and keep DAVAL = 1)
SETMWL	0x89			With two data bytes	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x8A			With two or three data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
GETMWL	0x8B	Read	No defining/sub-command byte	With two data bytes	X	X, GETF	Return data bytes from I3C_MAXWLR[15:0]. Refer to *Section 52.16.20*.
GETMRL	0x8C			With two or three data bytes	X	X, GETF	Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to *Section 52.16.19*.

Table 608. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
GETPID	0x8D	Read	No defining/sub-command byte	With six data bytes	X	X, GETF	Return data bytes from I3C_EPIDR. Refer to *Section 52.16.29*.
GETBCR	0x8E			With one data byte	X	X, GETF	Return data byte from I3C_BCR[7:0]. Refer to *Section 52.16.24*.
GETDCR	0x8F				X	X, GETF	Return I3C_DCR[7:0]. Refer to *Section 52.16.25*.
GETSTATUS	0x90		With or without defining byte (TGTSTAT, PRECR)	With two data bytes (format 1 or format 2 with PRECR)	X	X, STAF if format 1 X, GETF if format 2	Return 2 data bytes, as detailed in *Section 52.9.9*.
GETACCCR	0x91		No defining/sub-command byte	With one data byte	X	X, CRUPDF	Return data byte from I3C_DEVR0.DA[6:0] with parity bit
GETMXDS	0x94		With or without defining byte (WRRDTURN, CRHDLY)	With two data bytes (format 1) or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)	X	X, GETF	Return data byte(s) from I3C_GETMXDSR. Refer to *Section 52.16.28*.
GETCAPS	0x95		With or without defining byte (TGTSTAT, CRCAPS)	With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)	X	X, GETF	Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to *Section 52.16.26*) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to *Section 52.16.27*)
D2DXFER	0x97	Write	With defining byte	With defining byte	X	-	-
SETXTIME	0x98		With sub-command byte	With sub-command byte	X		
GETXTIME	0x99	Read	No defining/sub-command byte	No defining/sub-command byte	X		

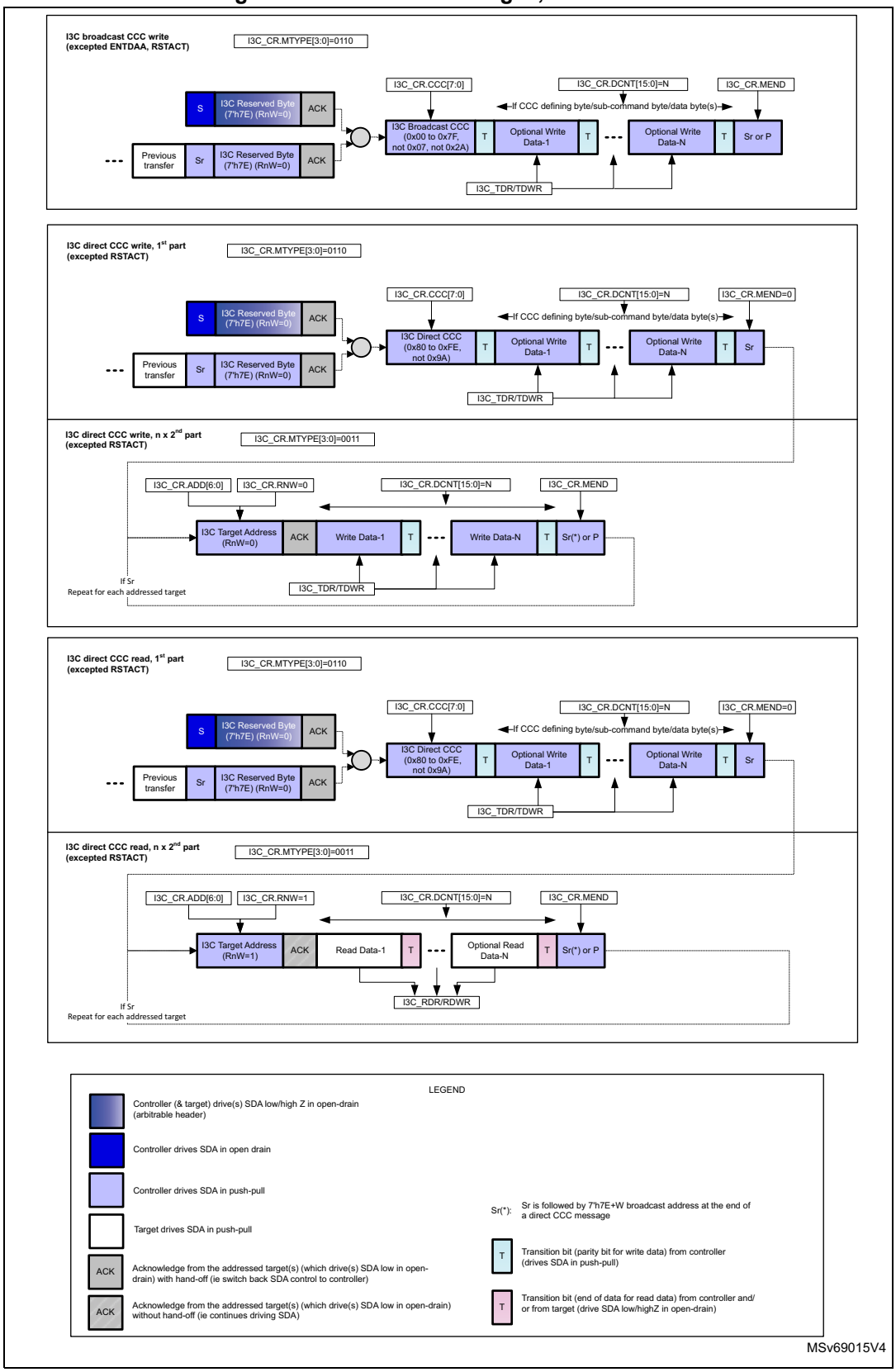
Table 608. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
RSTACT	0x9A	Read/ Write	With defining byte (0x00, 0x01, or 0x02)	With defining byte (0x00, 0x01, or 0x02)	X	X, RSTF if detected reset pattern	Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
SETGRPA	0x9B	Write	No defining/sub-command byte	No defining/sub-command byte	X	-	-
RSTGRPA	0x9C				X		

52.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

Figure 840 illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

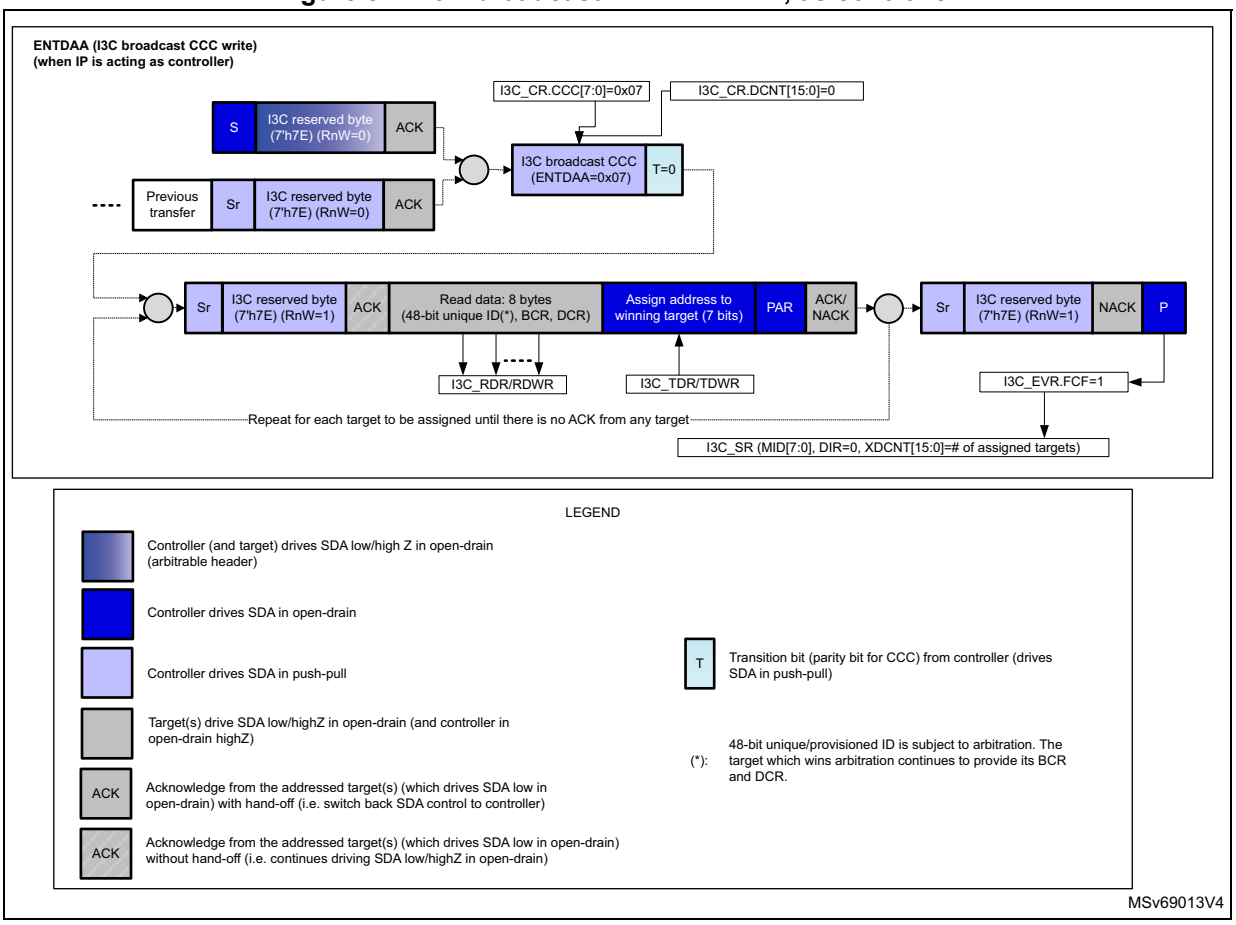
Figure 840. I3C CCC messages, as controller



52.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 841 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

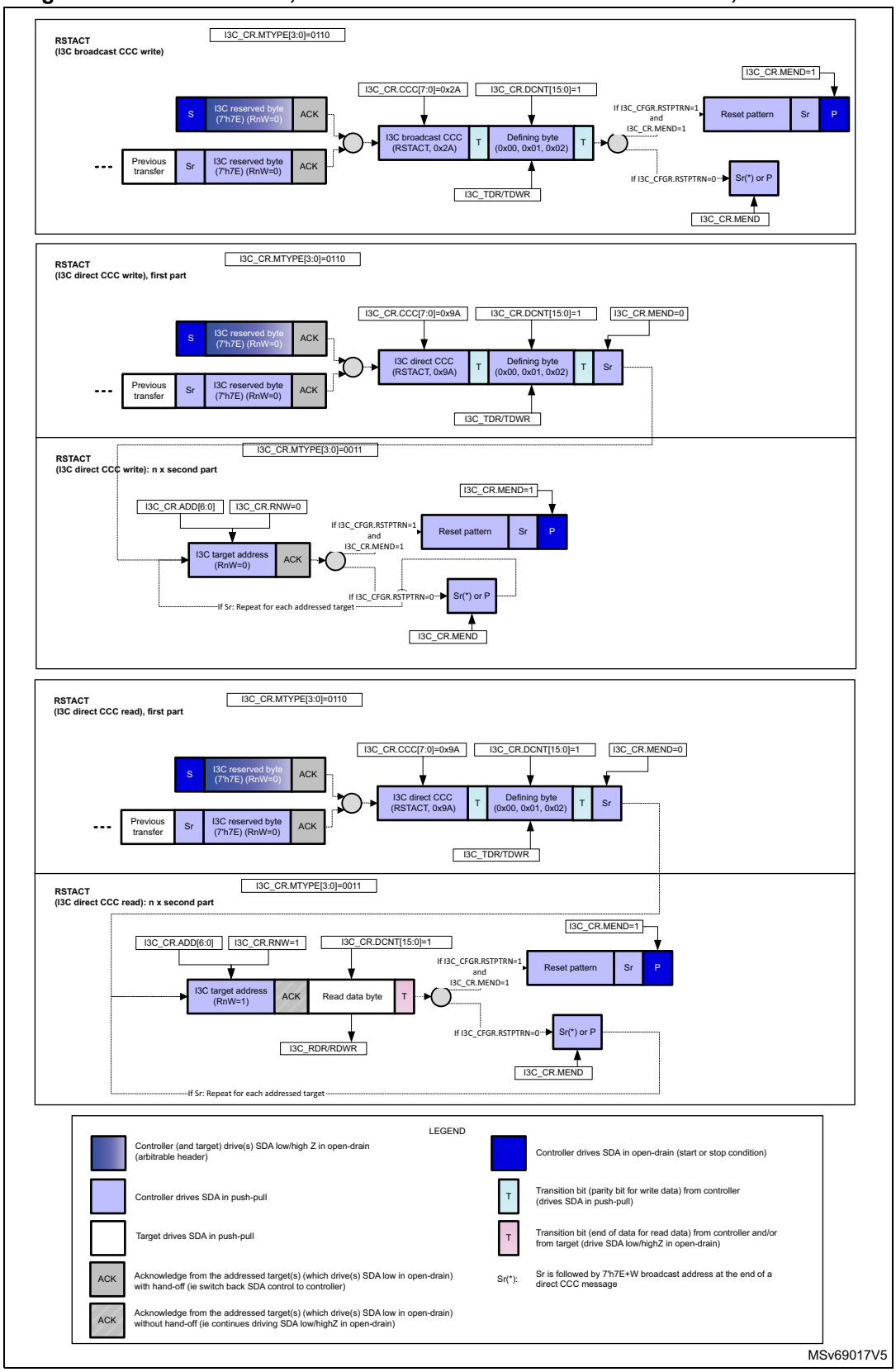
Figure 841. I3C broadcast ENTDAA CCC, as controller



52.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 842 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.

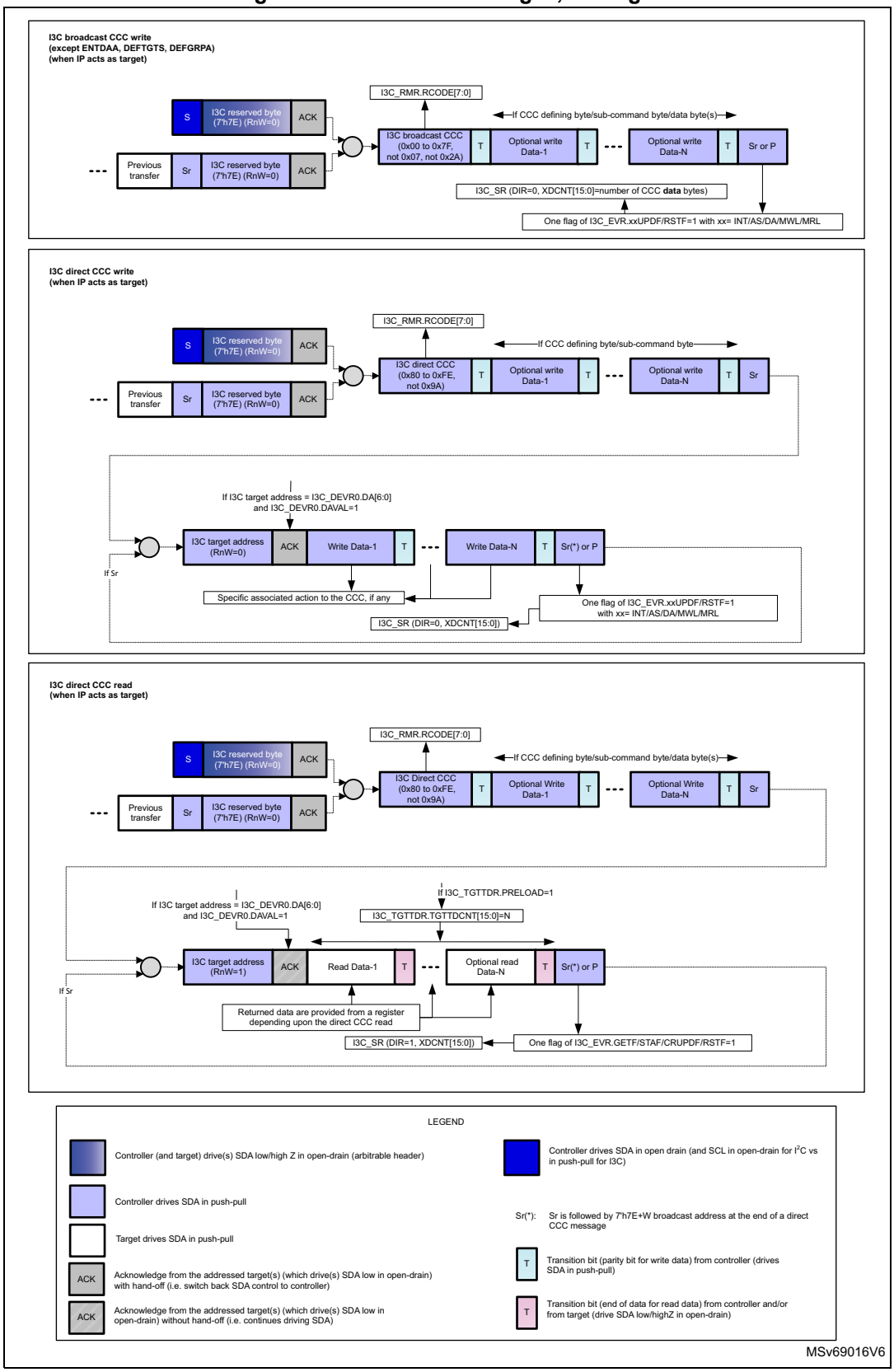
Figure 842. I3C broadcast, direct read and direct write RSTACT CCC, as controller



52.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target

Figure 843 illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGRPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

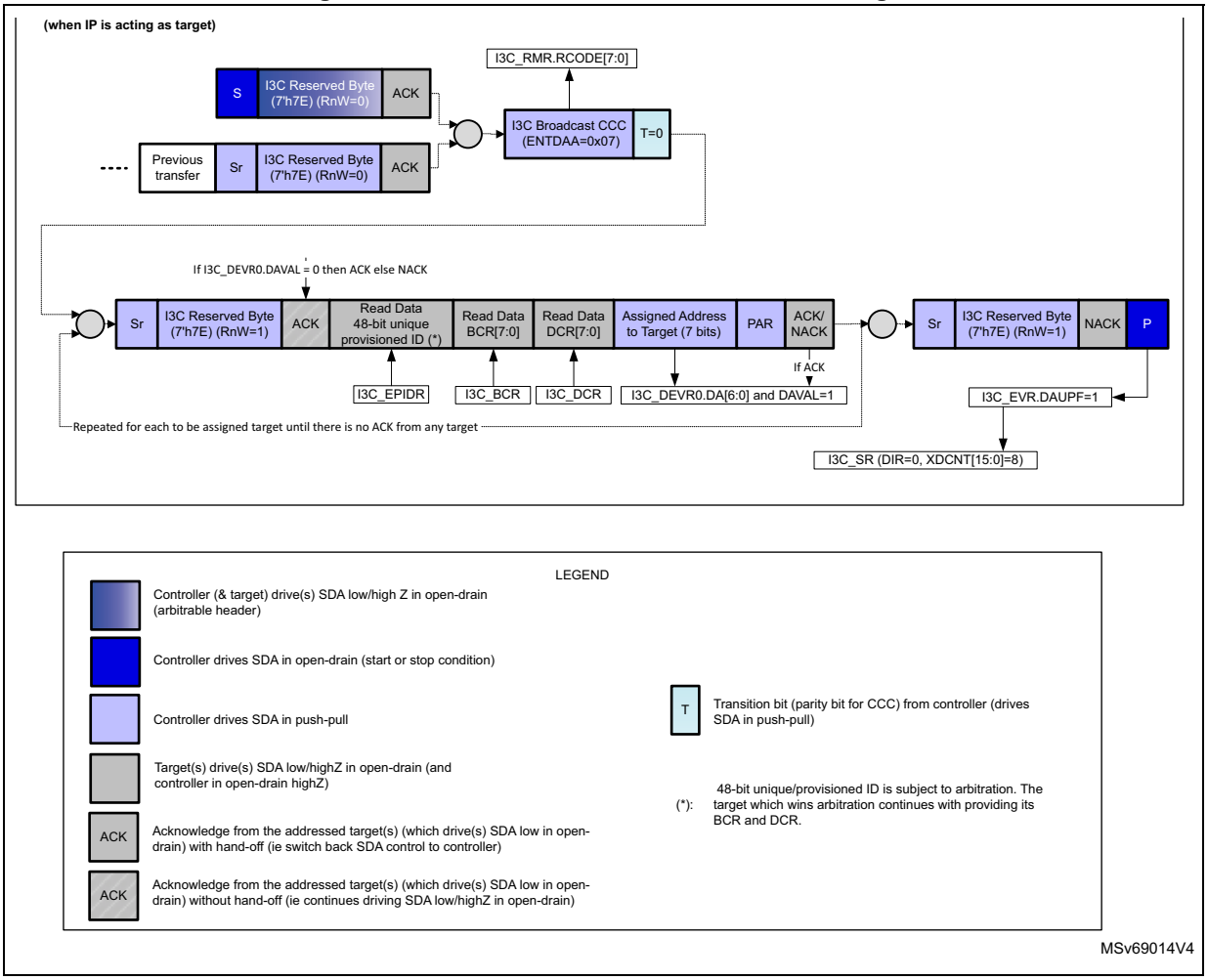
Figure 843. I3C CCC messages, as target



52.9.6 I3C broadcast ENTDAA CCC transfer, as target

Figure 844 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

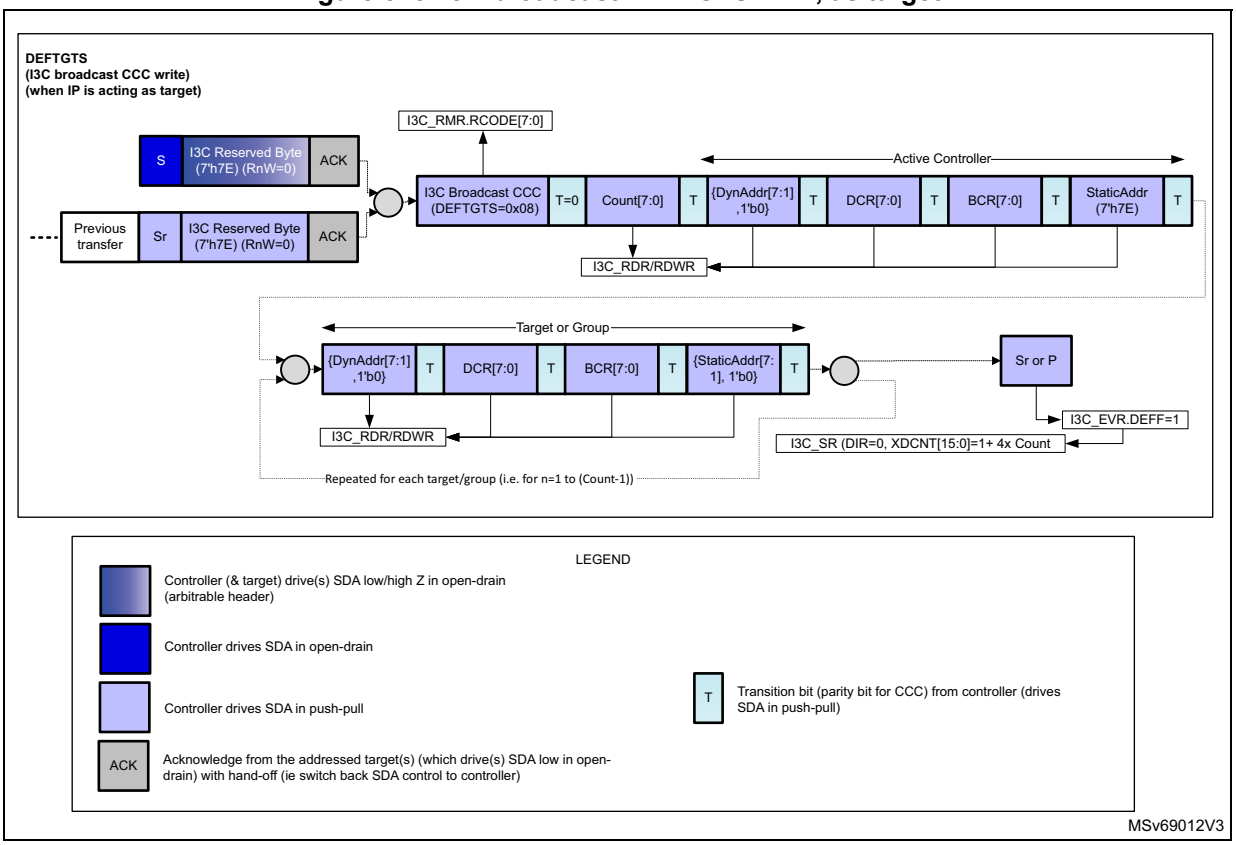
Figure 844. I3C broadcast ENTDAA CCC, as target



52.9.7 I3C broadcast DEFTGTS CCC transfer, as target

Figure 845 illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

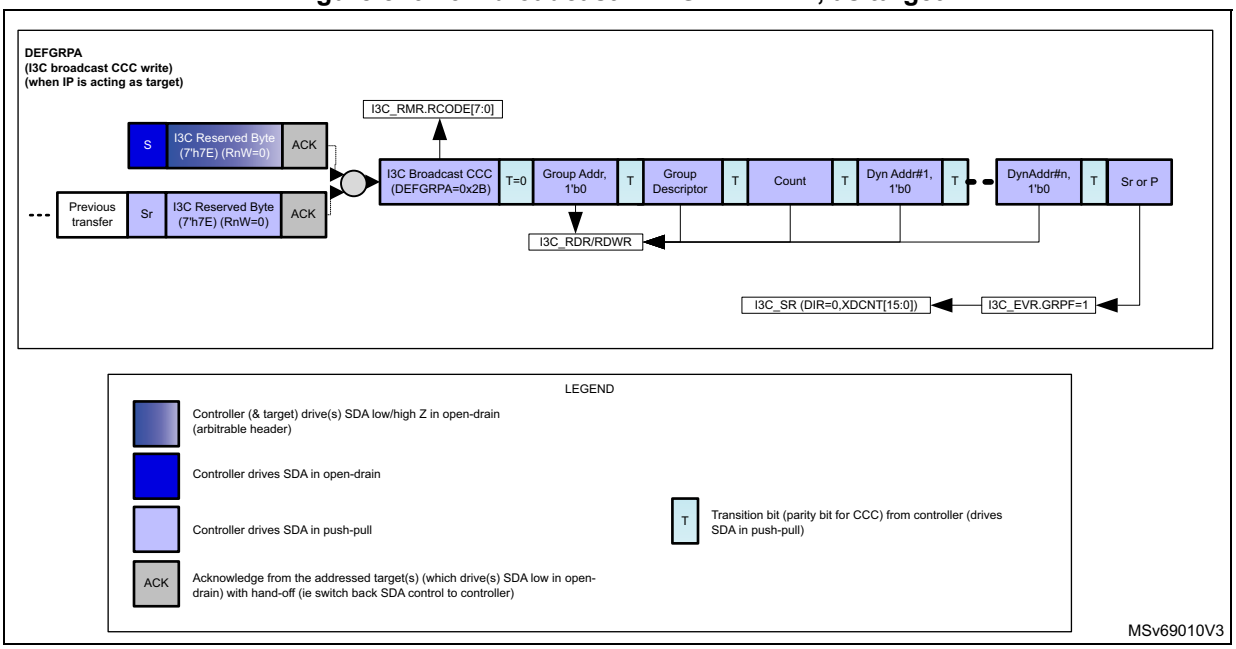
Figure 845. I3C broadcast DEFTGTS CCC, as target



52.9.8 I3C broadcast DEFGRPA CCC transfer, as target

Figure 846 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 846. I3C broadcast DEFGRPA CCC, as target



52.9.9 I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
- STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

- STATUS[15:8] = 0000 0000 (unused)
- STATUS[7:2] = 00000 (unused)
- STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
- STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

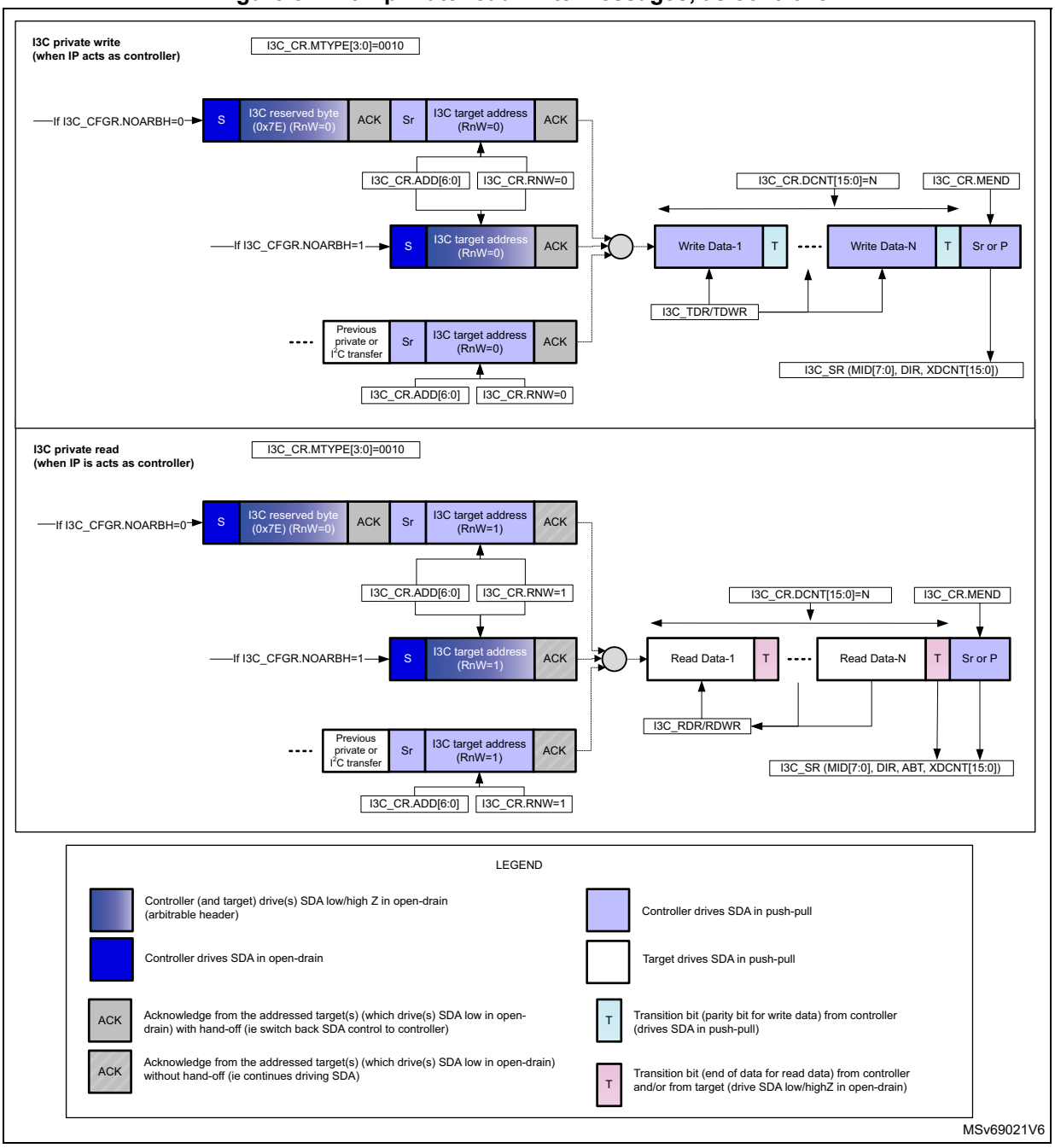
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).

52.9.10 I3C private read/write transfer, as controller

Figure 847 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

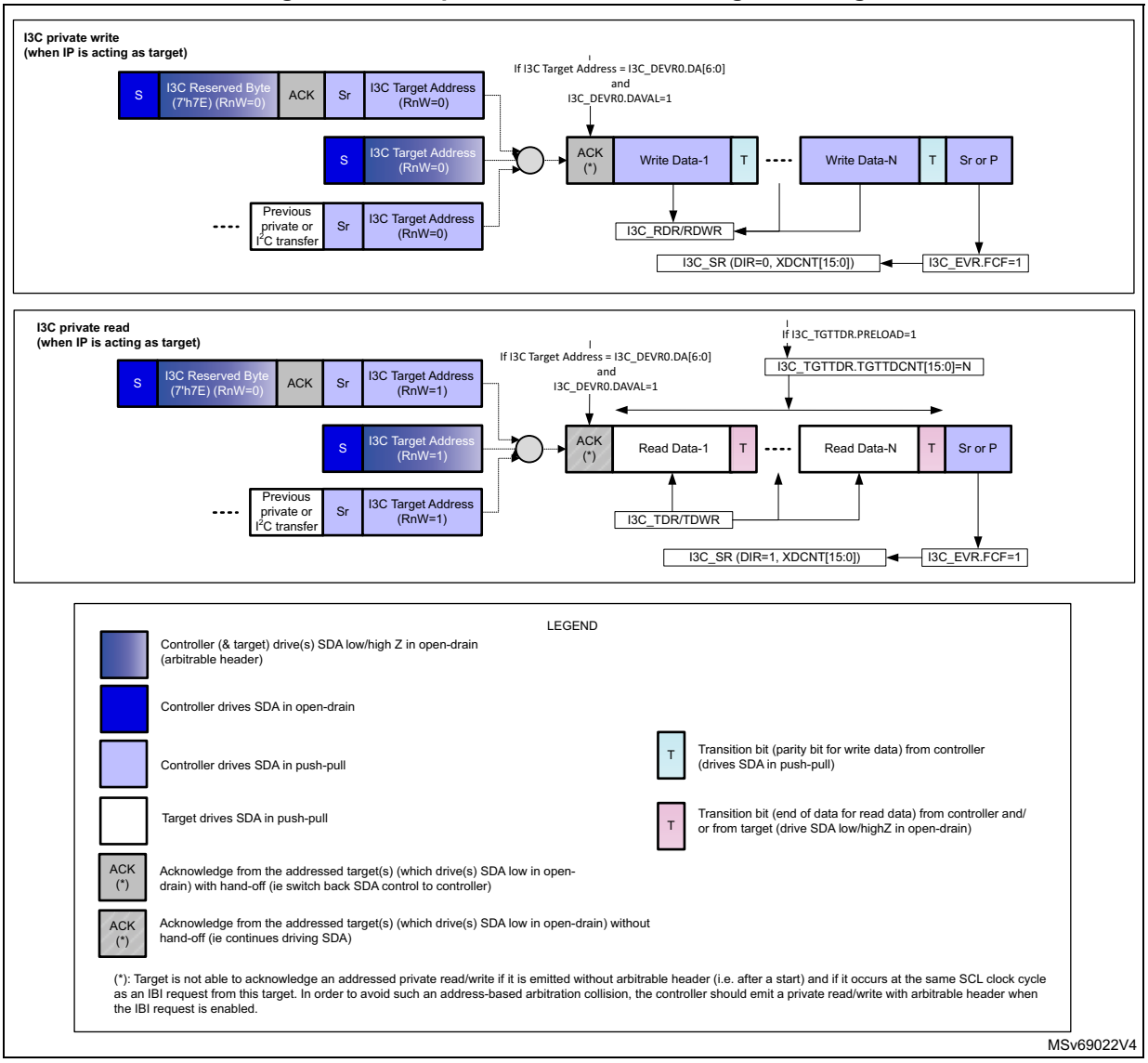
Figure 847. I3C private read/write messages, as controller



52.9.11 I3C private read/write transfer, as target

Figure 848 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

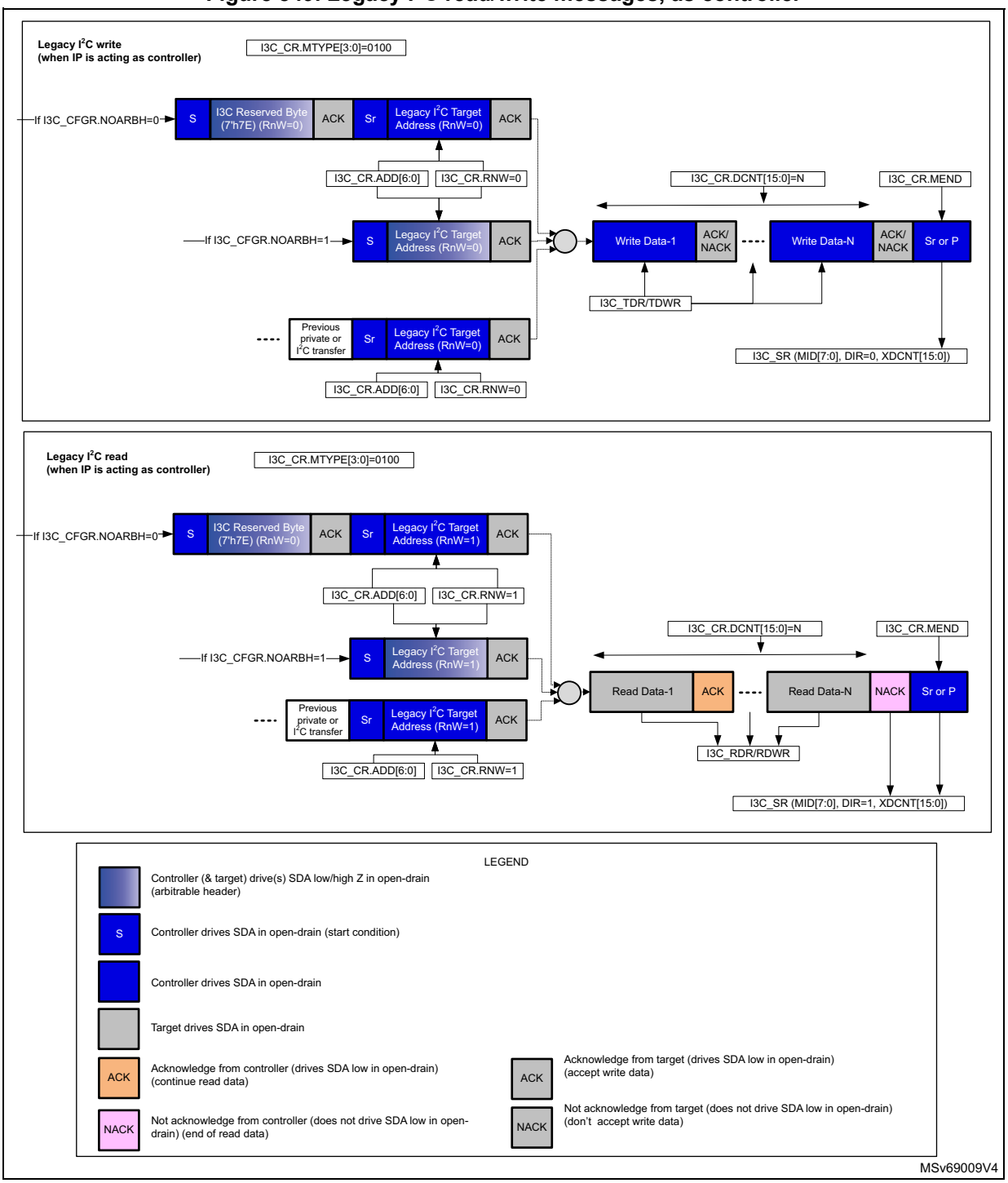
Figure 848. I3C private read/write messages, as target



52.9.12 Legacy I²C read/write transfer, as controller

Figure 849 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

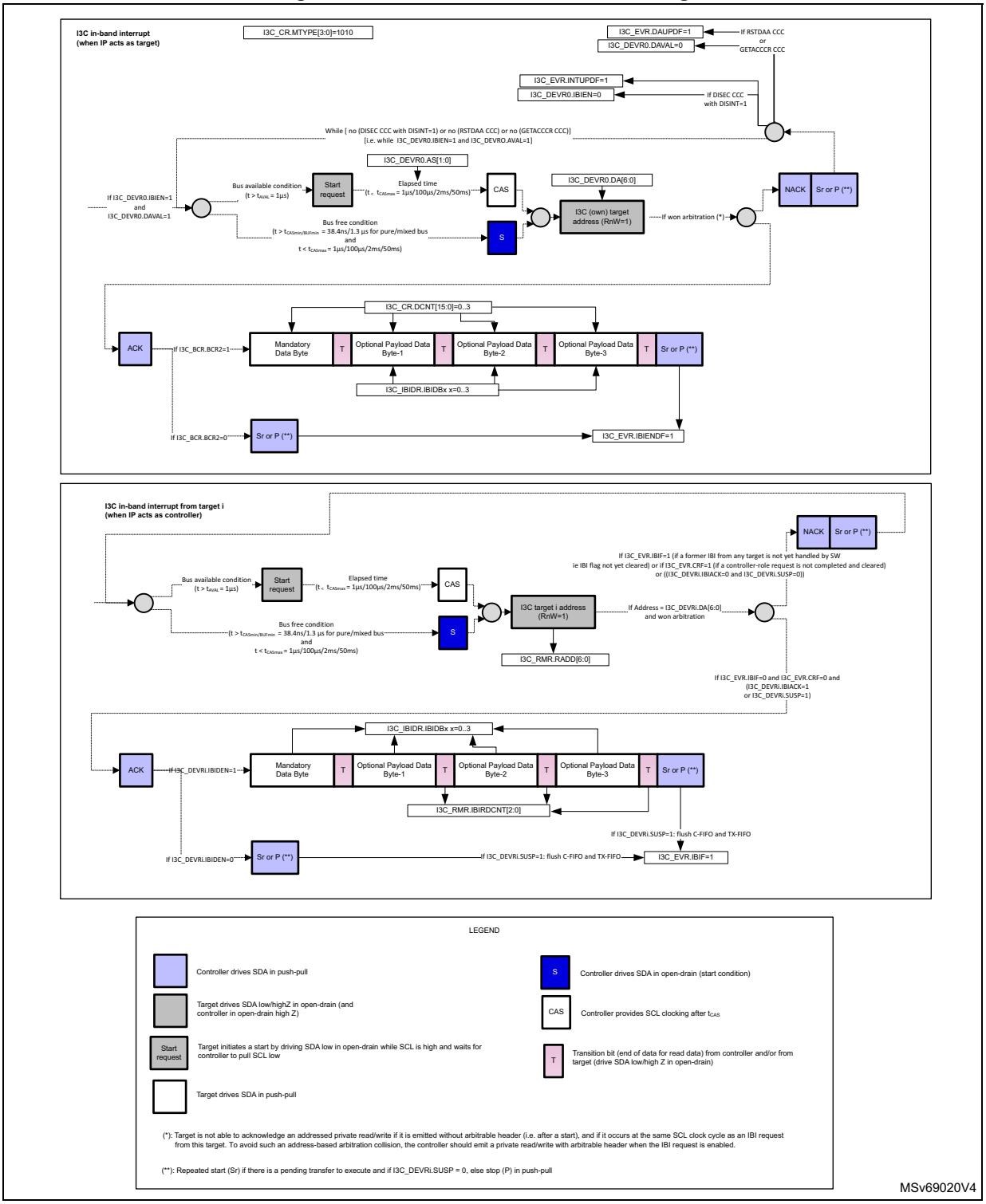
Figure 849. Legacy I²C read/write messages, as controller



52.9.13 I3C IBI transfer, as controller/target

Figure 850 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 850. IBI transfer, as controller/target



When the I3C peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

Timing control and sensor acquisition timestamping when I3C controller

When the I3C peripheral acts a controller, it can use timing control and asynchronous mode 0 to timestamp the moment at which the target(s) acquired sampled data, and to time-correlate them.

To do this, the software must go through the following sequence:

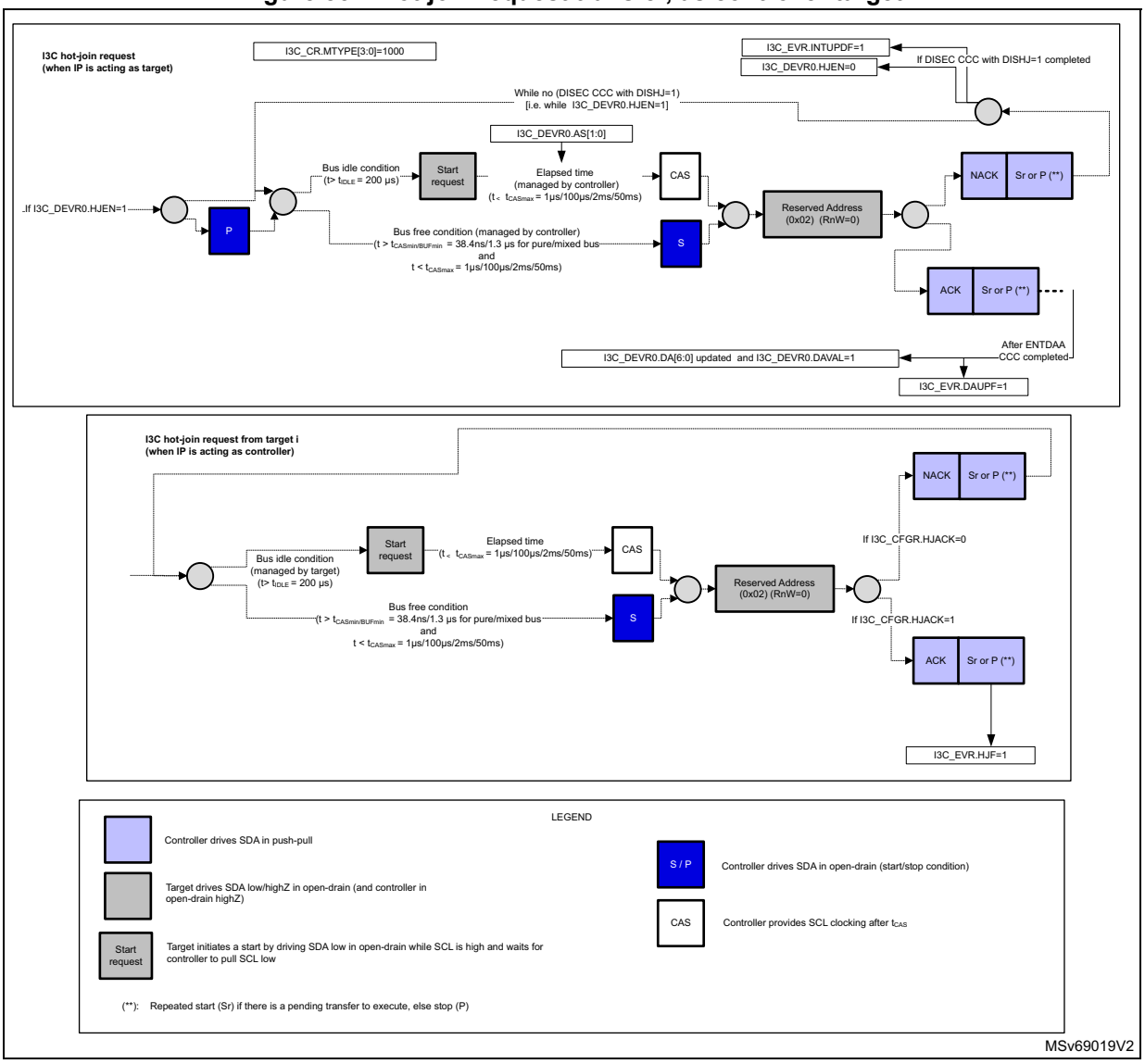
1. Allocate and configure a timer in capture mode, to be connected to the I3C generated event i3c_ibi_ack so that for each IBI acknowledge the counter of the timer can be captured.
2. Program a broadcast/direct SETXTIME CCC with sub-command byte 0xDT to enter asynchronous mode 0 on the target(s) side.
3. After being notified of the command completion by the flag and/or the related interrupt (FCF flag is set in I3C_EVR register), reset and enable the timer to start the counter on the controller side.
4. After being notified that an IBI is complete by the flag and/or the related interrupt (IBIF = 1 in the I3C_EVR register), and if the IBI is with timing info (if IBIDB0[7:5] = 100 in the I3C_IBIDR register), read the captured value of the timer, as C_TIM. Given that:
   a) C_C2 = 9 × (SCLL_PP[7:0] + 1 + SCLH_I3C[7:0] + 1) × T_{I3CCLK}
   b) T_C1 (16-bit value) = {IBIDB2, IBIDB1} in the I3C_IBIDR register
   c) T_C2 (8-bit value) = IBID3 in the I3C_IBIDR register

   Compute the timestamp of the sampled data received from the target, as follows (see MIPI spec formula for more details): C_TS = C_TIM - C_C2 x T_C1 / T_C2.
5. Repeat step *4*, for each received IBI.
6. Generate a broadcast/direct SETXTIME CCC with subcommand byte 0xFF to exit asynchronous mode 0 and disable/deallocate the timer resource.

52.9.14 I3C hot-join request transfer, as controller/target

Figure 851 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

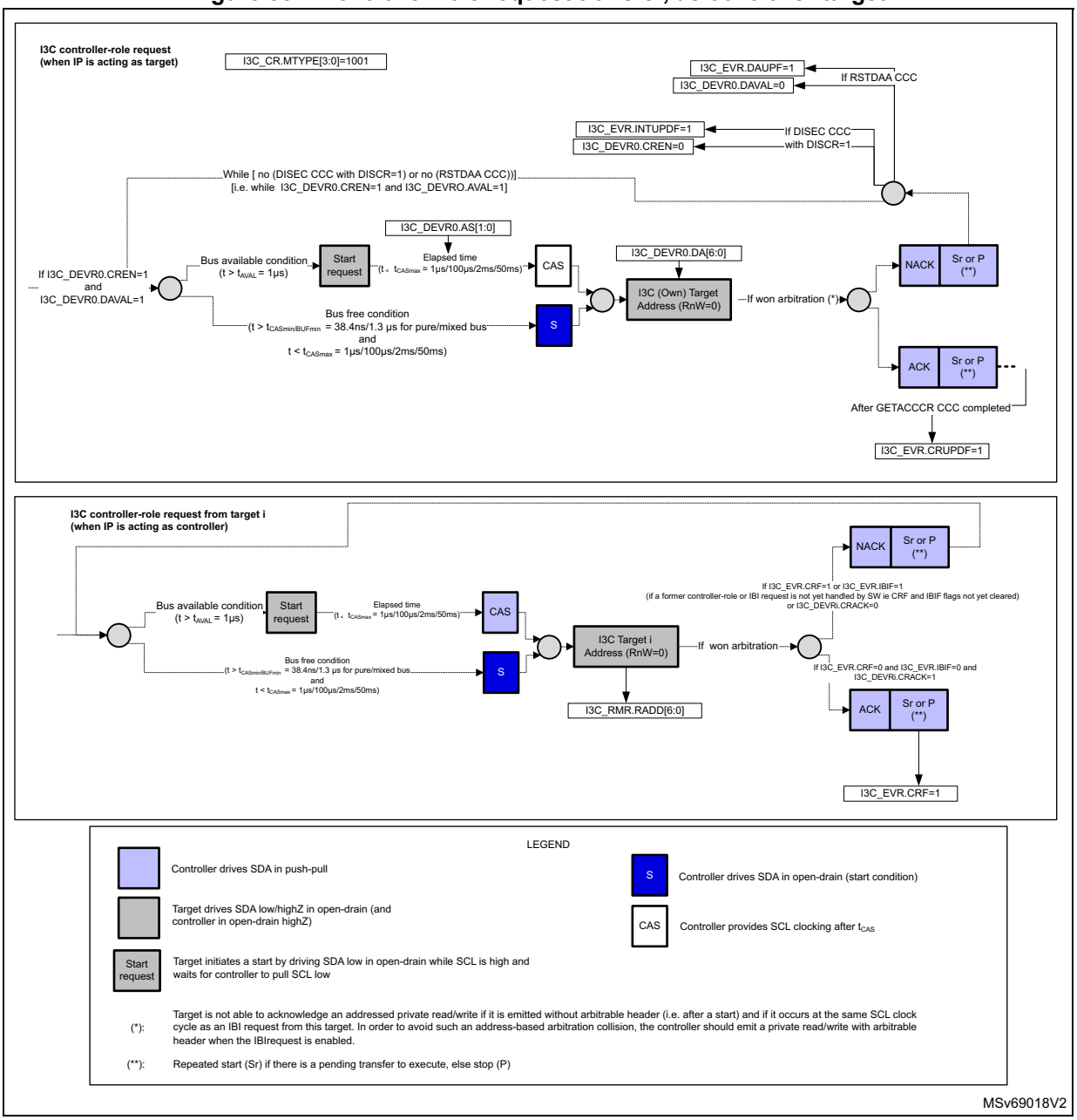
Figure 851. Hot-join request transfer, as controller/target



52.9.15 I3C controller-role request transfer, as controller/target

Figure 852 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 852. Controller-role request transfer, as controller/target



52.10 I3C FIFOs management, as controller

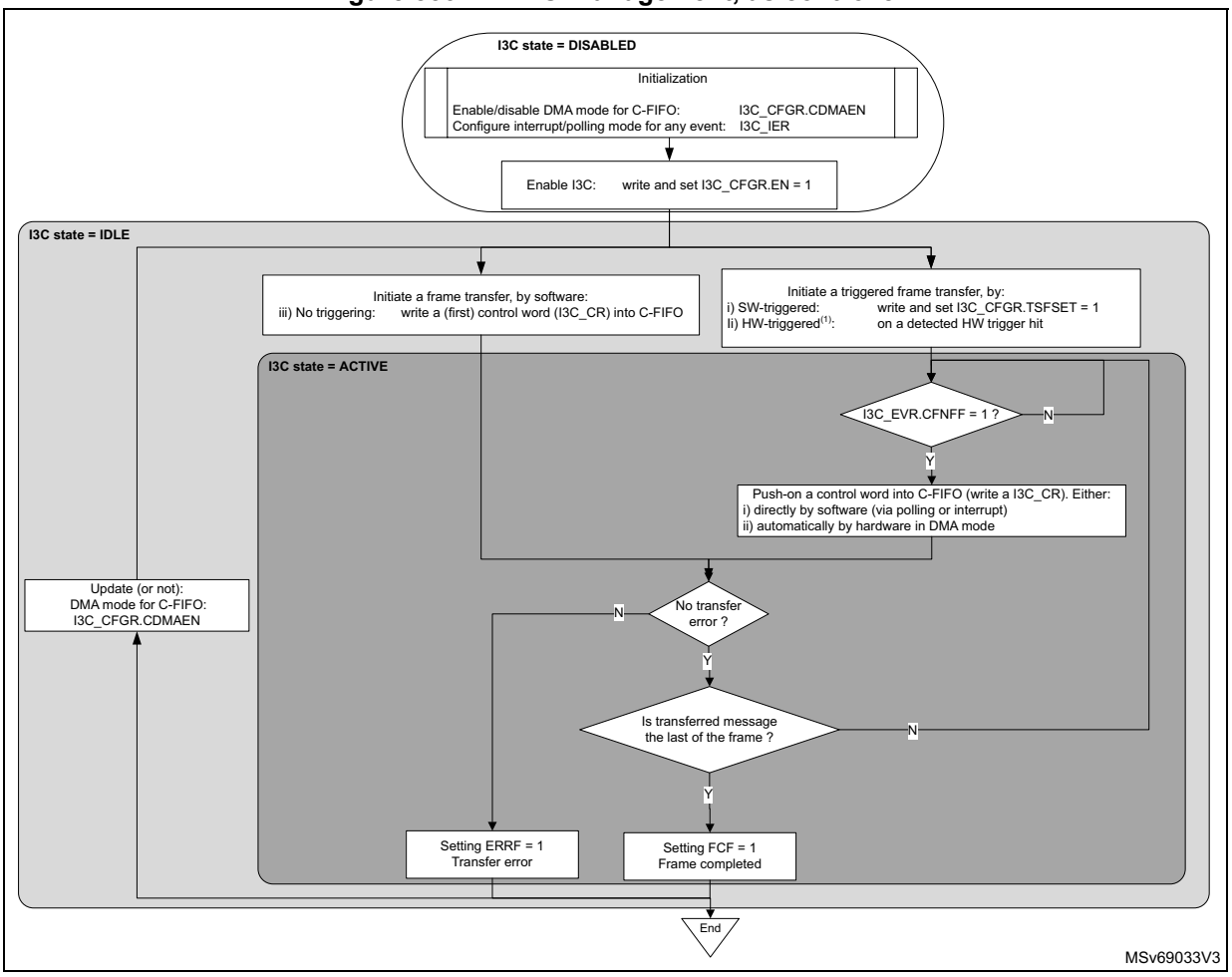
52.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of *Section 52.9*, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 840*, *Figure 841*, *Figure 842*)
- direct read/write CCC (*Figure 842*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 847*)
- legacy I2C read/write (*Figure 849*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 853 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

Figure 853. C-FIFO management, as controller



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 52.3.4*.

Note: The software can disable the generation of FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 52.3) to stream a series of frames.

First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

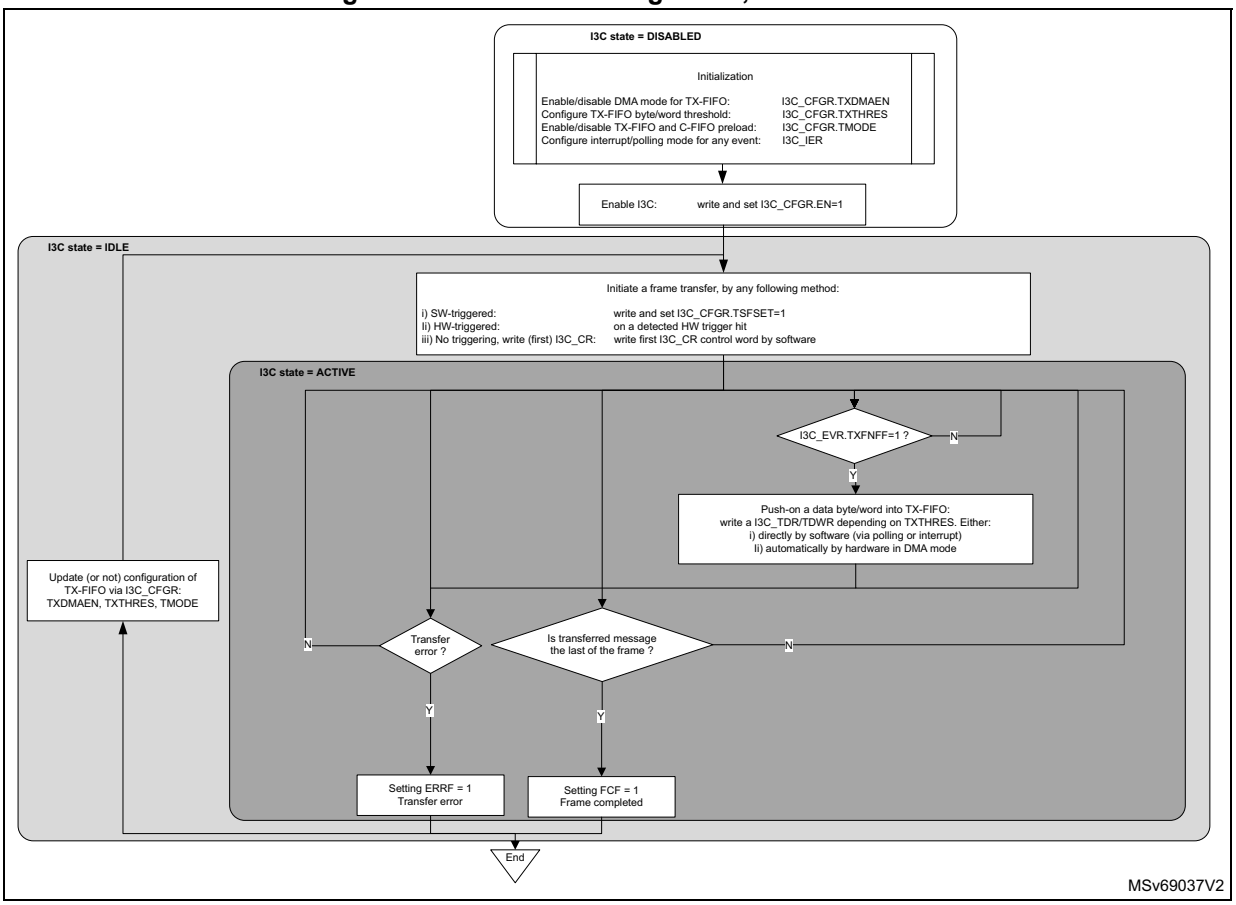
52.10.2 TX-FIFO management, as controller

When controller, as shown in figures of *Section 52.9*, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (*Figure 840*, *Figure 841*, *Figure 842*)
- private write (*Figure 847*)
- legacy I2C write (*Figure 849*)

Figure 854 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.

Figure 854. TX-FIFO management, as controller



First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) into TX-FIFO
- TMODE: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CGFR. register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in *Table 601*, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

1. Wait for a first control word to be written into the C-FIFO.
2. Wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.
3. If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
   a) Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
   b) If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control

word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

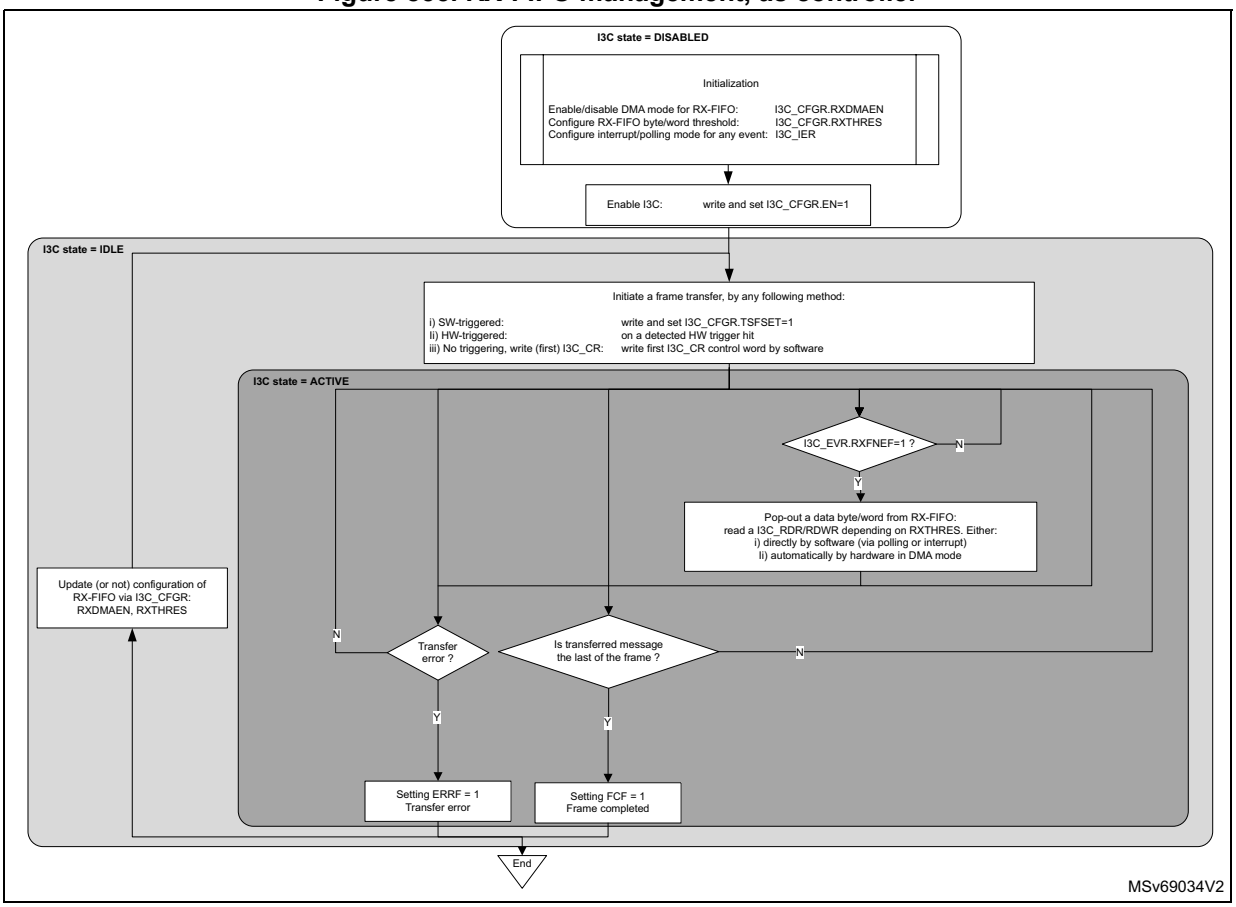
52.10.3 RX-FIFO management, as controller

When controller, as shown in figures of *Section 52.9*, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (*Figure 841*)
- direct CCC read (*Figure 840*), including direct RSTACT CCC read (*Figure 842*)
- private read (*Figure 847*)
- legacy I2C read (*Figure 849*)

Figure 855 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 855. RX-FIFO management, as controller



First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- If RXDMAEN = 1:
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- If RXDMAEN = 0:
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.

In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message ,and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to *I3C status register (I3C_SR)* and *Section 52.10.4*.

52.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the *I3C status register (I3C_SR)* for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferee a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

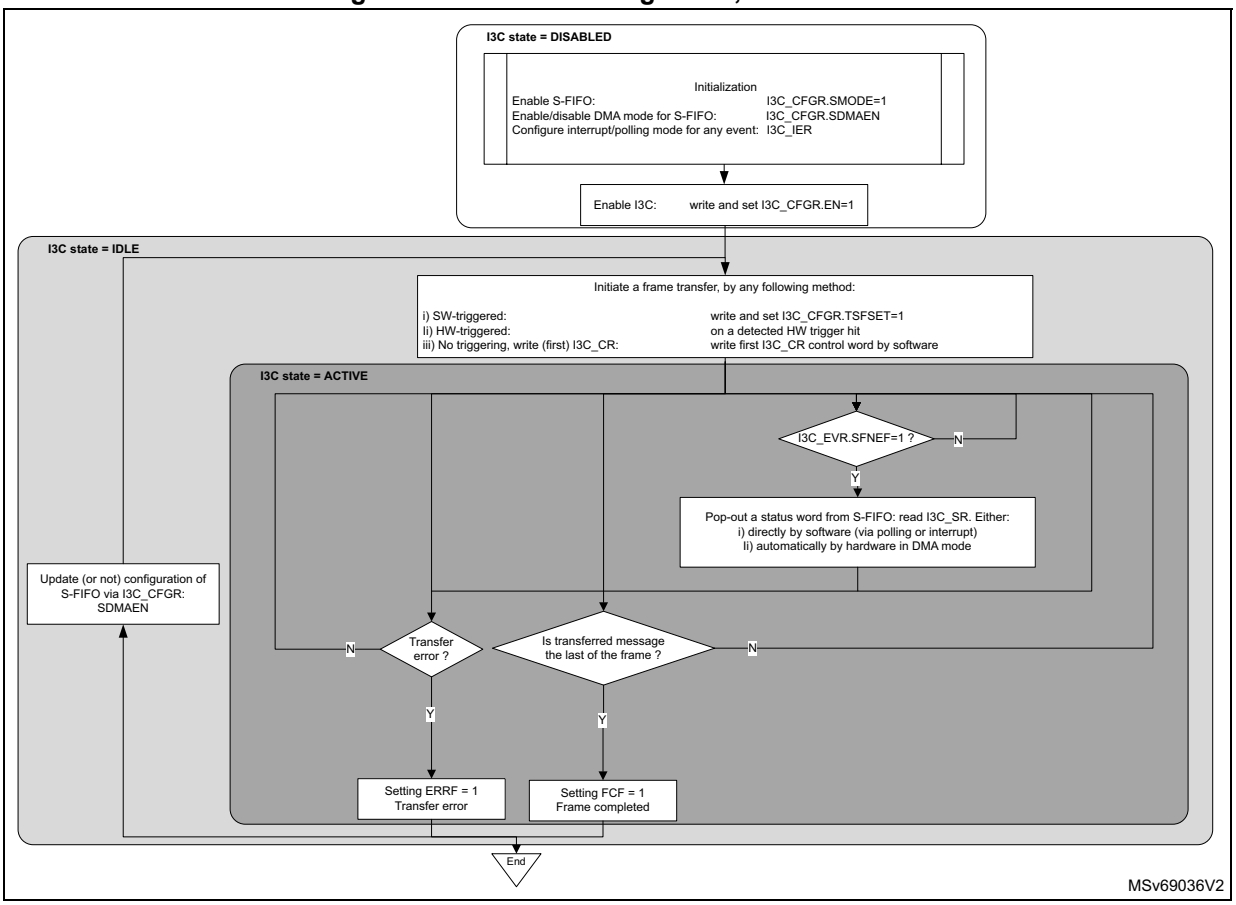
- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to *Section 52.10.3*).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. *Figure 856* illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.

Figure 856. S-FIFO management, as controller



First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.

The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

52.11 I3C FIFOs management, as target

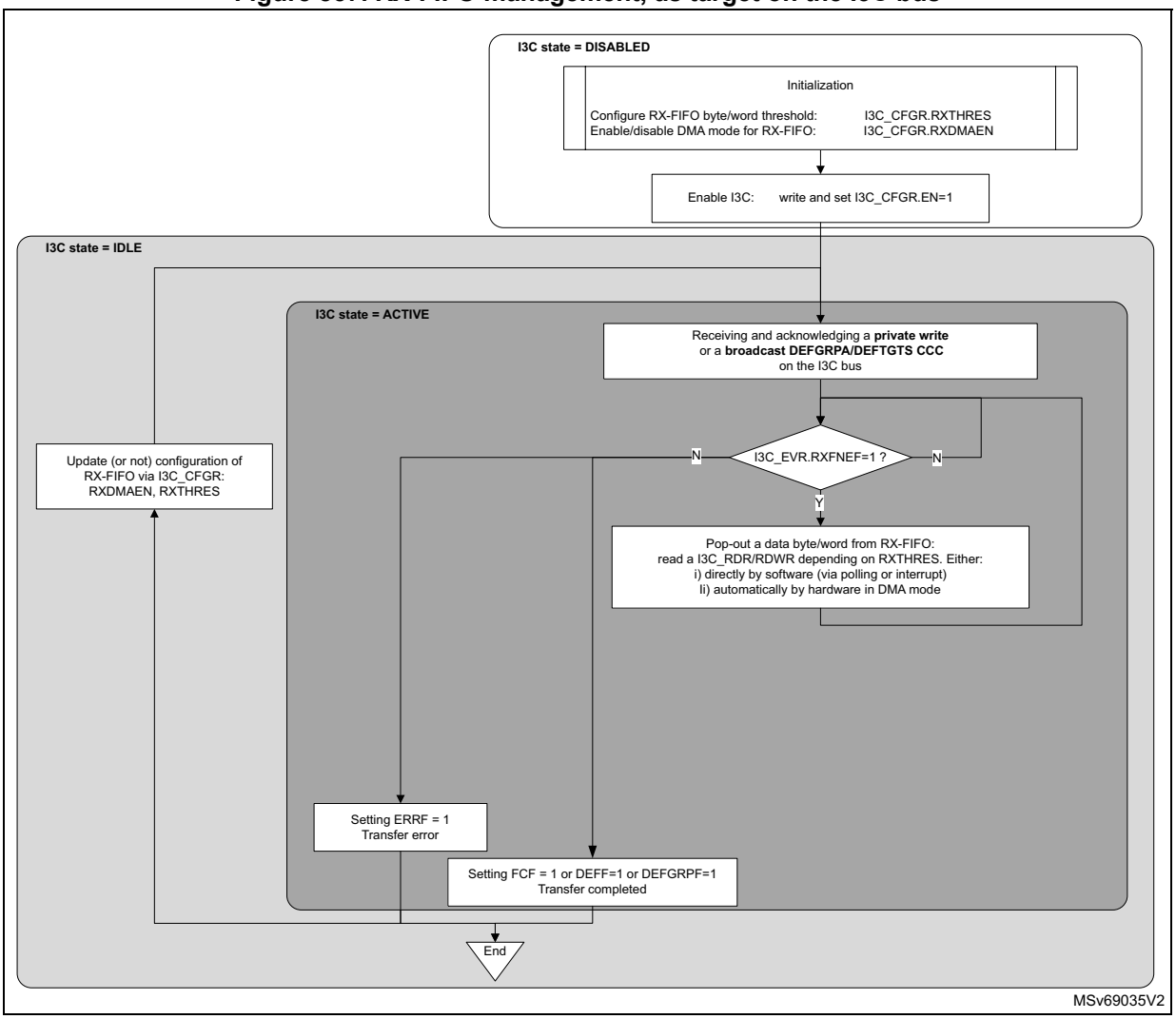
52.11.1 RX-FIFO management, as target

When target, as shown in figures of *Section 52.9*, the RX-FIFO is used during any of the following received and acknowledged transfers:

- broadcast DEFTGTS CCC (*Figure 845*)
- broadcast DEFGRPA CCC (*Figure 846*)
- private write (*Figure 848*)

Figure 857 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received on the I3C bus, when the I3C peripheral acts as target.

Figure 857. RX-FIFO management, as target on the I3C bus



First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed or private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

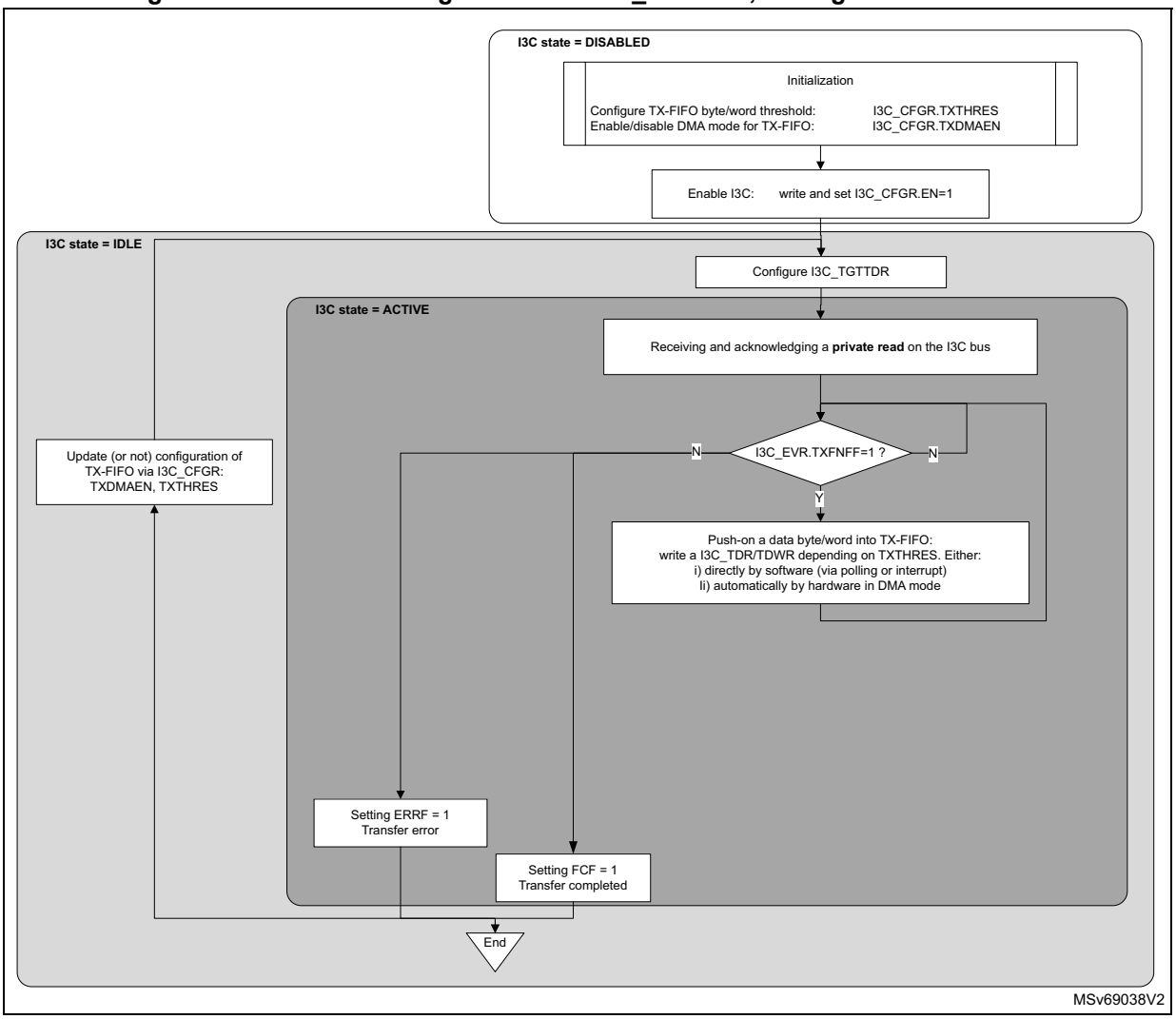
52.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of *Section 52.9*, the TX-FIFO is used only during a private read (*Figure 848*).

Figure 858 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.

Figure 858. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus



First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus , the software must configure the *I3C target transmit configuration register (I3C_TGTTDR)* to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the serial bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 601).

Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the serial bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the serial bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the serial bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGR register, continue (or not) for another read, depending upon the user application

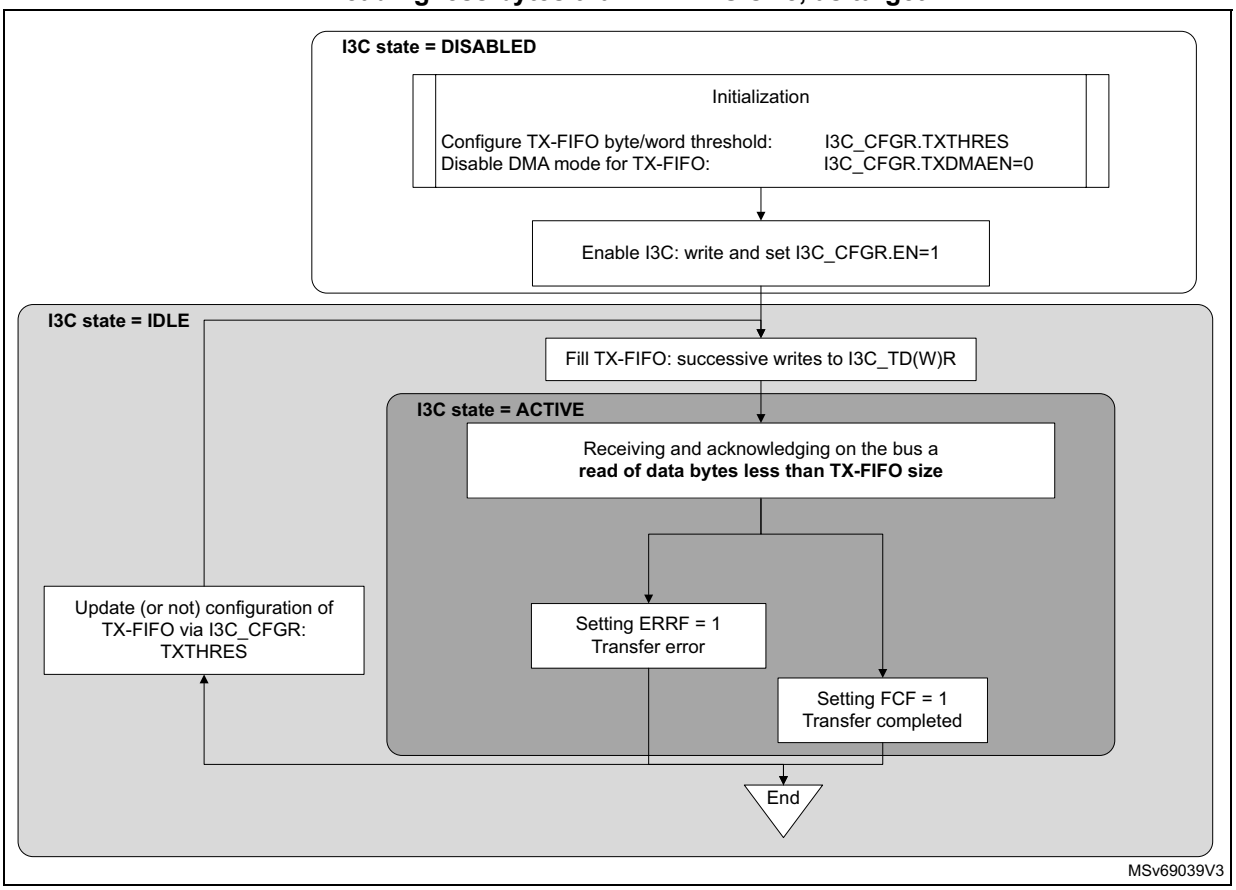
If TX-FIFO is empty and a data byte must be transmitted on the serial bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size

Alternatively to the use of the I3C_TGTTDR register, as shown in the *Figure 859*, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the serial bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.

Figure 859. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target



52.12 I3C error management

52.12.1 Controller error management

Table 609 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

Table 609. I3C controller error management

Error type	Description	Error detection	Controller action	Reported error(1)
CE0	Illegally formatted CCC (e.g. less returned data byte(s))	Prematurely ended read data by target on a direct CCC read(2)	Hardware emits a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000

Table 609. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
CE1	Monitoring error	An incorrect ACK is detected at the end of a legacy I^2C read	Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001
		Restart or stop cannot be generated at the end of an I3C SDR read	SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.	
		After a CE1 error, a start cannot be generated		
CE2	No response to broadcast address (0b111_1110)	Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message	Hardware emits an HDR exit pattern followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010
CE3	Failed controller hand-off	New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed	Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011
-	Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC	Incorrect dynamic address and/or parity bit is detected	Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and DERR = 1
-	Addressed target is NACK-ed on a direct CCC read for the first time	NACK is detected	Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.	-
-	Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I^2C, or a direct CCC read for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and ANACK= 1

Table 609. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time	NACK is detected	Hardware performs a single-retry assignation loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.	-
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and DNACK= 1
-	Write data is NACK-ed on a legacy I2C write			
-	Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings	SCL stall timeout	Hardware emits a stop.	ERRF = 1 and (COVR = 1 or DOVR = 1)

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4.
On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

52.12.2 Target error management

Table 610 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 610. I3C target error management

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE0	Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment	A forbidden address is detected after a start or a repeated start	Hardware waits for an HDR exit pattern	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000
TE1	CCC code	A CCC code is detected with a parity error		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001

Table 610. I3C target error management (continued)

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE2	Write data	A write data byte is detected with a parity error on an I3C private write message	Hardware waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1010
TE3	Assigned address during dynamic address arbitration	Assigned dynamic address is detected with a parity error on a ENTDAA CCC		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1011
TE4	0b111_1110 + RnW = 1 missing after Sr during dynamic address arbitration	Missing {0b111_1110 + RnW = 1} after Sr during dynamic address arbitration is detected		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1100
TE5	Transaction after detecting CCC	Invalid CCC direction is detected vs. direction provided during address phase, on a CCC direct read/write		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1101
TE6	Monitoring error of I3C bus	SDA is detected as driven at unexpected value on an data read (during a CCC direct read or a private read or an IBI)	Hardware releases SDA, then waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1110
-	Monitoring SCL on an SDR data read	SCL is detected as stable for more than 125 µs on an SDR data read (during a CCC direct read or a private read or an IBI)		ERRF = 1 and STALL = 1
-	Data to be transmitted not written/available in time vs. I3C bus timings	Data to be transmitted not written by software or DMA in due time on a private read	Hardware detects a stop or a repeated start	ERRF = 1 and DOVR = 1
-	Received data cannot be registered in time vs. I3C bus timings	Received data is not read by software or DMA in due time on a private write or a DEFTGTS or a DEFGRPA CCC		

1. ERRF in the I3C_EVR register, PERR, ODERR[3:0], STALL, and DOVR in the I3C_SER register.

52.13 I3C wake-up from low-power mode(s)

52.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to Section Reset and clock control (RCC).

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.

When entering a Stop mode, the V_{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay defines the T_{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx with the value x = 1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T_{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.
   - On a detected hardware trigger to initiate an I3C transfer on the I3C bus (refer to *Section 52.3.4: I3C triggers* to determine whether the trigger feature is supported).
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.
3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 850*), and it maintains the APB clock request until that IBIF is cleared.

     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 851*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 852*), and it maintains the APB clock request until that HJF is cleared.
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as

the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.

7. With the APB clock running, the I3C peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the I3C peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

As target: wake-up on a I3C reset pattern

The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the I3C peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start on the serial bus

1. The I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the serial bus (SDA line is detected to be driven low while SCL is high)
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a bus start is missed and waits for the kernel clock to be provided. A controller message that is specifically addressed to this low-power device may have been missed (and NAcked). However the I3C peripheral can also detect a missed start if the controller started a CCC or private or legacy I^2C message to another addressed target, or if another target initiated a start for an IBI/CR/HJ request.
   - On a detected hardware trigger to initiate an I3C IBI/CR/HJ transfer on the I3C bus (refer to *Section 52.3.4: I3C triggers* to determine whether the trigger feature is supported).
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in

Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).

3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   – The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that a serial bus transaction was missed, but not whether the target was addressed or not. The software uses a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.

52.14 I3C in low-power modes

Table 611. Effect of low-power modes

Mode	Description
Sleep	No effect. I3C interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode.
Standby	The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 52.3.2: I3C wake-up from low-power mode(s)* to see which, if any, Stop modes are supported.

52.15 I3C interrupts

52.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

52.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description *Section 52.16.2* must be considered.
- when I3C acts as target (bit[30] = MTYPE[3] = 1).

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0]							RNW
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: this message from controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (whatever I3C acts as controller/target)

Condition: when I3C acts as I3C controller

0000: **SCL clock is forced to stop** until a next control word is executed

Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL “stuck at” recovery. Refer to *Table 609*.

0001: **header message**

Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to *Table 609* and MIPI specification about escalation handling.

If the fields of the I3C_CFGR register are programmed with EXITPTRN= 0 and RSTPTRN=1, a target reset pattern is sent on the I3C bus after the header message, whatever the header is ACK-ed or NACK-ed. This can be used to generate and broadcast a target reset pattern, especially in the case of an I3C bus topology with a single I3C target in deep sleep. Alternatively, and more generally, a target reset pattern can be encapsulated within a RSTACT CCC command (refer to *Figure 842*).

0010: **private message** (refer to *Figure 847*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred private message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
– After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).

0011: **direct message (second part of an I3C SDR direct CCC command)** (refer to *Figure 840*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P

0100: **legacy I^2C message** (refer to *Figure 849*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred legacy I^2C message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
– After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

Condition: when I3C acts as I3C target

1000: **hot-join request** (W) (refer to *Figure 851*)

The transferred hot-join request is {S +} 0b000_0010 addr + RnW = 0.
Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: **controller-role request** (W) (refer to *Figure 852*)

The transferred controller-role request is {S +} DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register)
Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: **IBI (in-band interrupt) request** (R) (refer to *Figure 850*)

Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
The transferred IBI request is {S +} DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:17 **ADD[6:0]**: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 **RNW**: Read / non-write message (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.

0: write message

1: read message

Bits 15:0 **DCNT[15:0]**: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.

When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).

Linear encoding up to 64 Kbytes - 1

0x0000: no data to transfer

0x0001: 1 byte

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

52.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to *I3C message control register (I3C_CR)*.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	CCC[7:0]							
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

Condition: when I3C acts as I3C controller

0110: broadcast/direct CCC command (refer to *Table 608*, *Figure 840*, *Figure 841*, *Figure 842*)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code

Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.

If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command

The transferred direct CCC command (first part) message is:

– {S / S + 0b111_1110 +RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr
– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to *Figure 841*)

The transferred broadcast CCC command message is:

– {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P
– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.

If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

Linear encoding up to 64 Kbytes - 1.

0x0000: no data to transfer.

Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 841).

0x0001: 1 byte

Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 842).

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

52.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH, FCFDIS)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	FCFDIS	Res.	CFLUSH	CDMA EN	TMODE	SMODE	SFLUSH	SDMA EN
	w							rw		w	rw	rw	rw	w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX THRES	TX FLUSH	TX DMAEN	Res.	RX THRES	RX FLUSH	RX DMAEN	HJACK	Res.	HK SDAEN	EXIT PTRN	RST PTRN	NO ARBH	CRINIT	EN
	rw	w	rw		rw	w	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TSFSET**: Frame transfer set (software trigger) (when I3C acts as controller)

This bit can only be written. When I3C acts as I3C controller:

0: no action

1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed)

Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 **FCFDIS**: FCF generation disable (when I3C acts as controller)

When the IP acts as controller, the generation of the FCF flag in the register I3C_EVR can be disabled. The software can avoid clearing this flag between two generated I3C frames (by writing 1 into CFCF in the I3C_CEVR register), especially in hardware-triggered autonomous mode.

0: enable the generation of the flag FCF in the I3C_EVR register

1: disable the generation of the flag FCF in the I3C_EVR register

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CFLUSH**: C-FIFO flush (when I3C acts as controller)

This bit can only be written.

0: no action

1: flush C-FIFO

Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:

0: DMA mode is disabled for C-FIFO

- Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
- A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).

1: DMA mode is enabled for C-FIFO

- DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
- A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.

0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.

A frame transfer starts as soon as the first control word is present in C-FIFO.

1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to *Section 52.10.2* for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.

When I3C acts as target, this bit must be cleared.

0: S-FIFO is disabled

- Status register (I3C_SR) is used without FIFO mechanism.
- There is no SCL stalling if a new status register content is not read.
- Status register must be read before being overwritten by the hardware.
- Must have SDMAEN = 0 in the I3C_CFGR register.

1: S-FIFO is enabled.

- Each message status must be read.
- There is SCL stalling when the S-FIFO is full and a next message status must be read.
- S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.

0: no action

1: flush S-FIFO

Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFGR register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR

- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR

- Must have SMODE = 1 in the I3C_CFGR register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold

TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold

TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action

1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO

- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO

- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold

RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold

RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action

1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO

- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO

- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged

After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged

After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN:** High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled

1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

Note: This bit can be modified only when EN = 0 in the I3C_CFGR register.

Bit 4 **EXITPTRN:** HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header {S/Sr + 0x7E addr + W} is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN:** HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Additionally, if RSTPTRN = 1 and EXITPTRN = 0, a reset pattern is sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register), whatever the message header is ACK-ed or NACK-ed.

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I^2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I^2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to ensure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the I3C peripheral is under reset (partial reset).
- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.
- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFGR)

52.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RDB0[7:0]**: 8-bit received data on I3C bus.

52.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDB3[7:0]								RDB2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDB1[7:0]								RDB0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).

Bits 23:16 **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).

Bits 15:8 **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).

Bits 7:0 **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

52.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

52.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDB3[7:0]								TDB2[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDB1[7:0]								TDB0[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8 **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0 **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)

52.16.8 I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:

- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:

- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBIDB3[7:0]								IBIDB2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIDB1[7:0]								IBIDB0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IBIDB3[7:0]**: 8-bit IBI payload data (latest byte on I3C bus).

Bits 23:16 **IBIDB2[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).

Bits 15:8 **IBIDB1[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).

Bits 7:0 **IBIDB0[7:0]**: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).

52.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024

Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFGR register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGR register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGTTDCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)

This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.

This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.

0: no TX-FIFO preload

1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)

This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.

This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.

52.16.10 I3C status register (I3C_SR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.
r	r	r	r	r	r	r	r						r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XDCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **MID[7:0]**: Message identifier/counter of a given frame (when the I3C acts as controller)

When I3C acts as controller, this fbitield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.

First message of a frame is identified with MID[7:0] = 0.

This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **DIR**: Message direction

Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus

0: write

1: read

Note: ENTDAA CCC is considered as a write command.

Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target

1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)

- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message

- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

52.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error

1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error

1: controller detected that a data byte is not acknowledged by a target, either during:

i) a legacy I2C write transfer

ii) the second trial when sending dynamic address during ENTDAA procedure

Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)

0: no detected error

1: controller detected that the static/dynamic address was not acknowledged by a target, either during:

i) a legacy I2C read/write transfer

ii) a direct CCC write transfer

iii) the second trial of a direct CCC read transfer

iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)

0: no detected error

1: controller detected either:

i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted

ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun

0: no detected error

1: whatever controller or target, hardware detected either:

i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted

ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)

0: no detected error

1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error

0: no detected error

1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]

Bits 3:0 **CODERR[3:0]**: Protocol error code/type
0000: CE0 error (transaction after sending CCC):
controller detected an illegally formatted CCC
0001: CE1 error (monitoring error):
controller detected that transmitted data on the bus is different from expected
0010: CE2 error (no response to broadcast address):
controller detected a not acknowledged broadcast address (0b111_1110)
0011: CE3 error (failed controller-role hand-off):
controller detected the new controller did not drive bus after controller-role hand-off
1000: TE0 error (invalid broadcast address 0b111_1110 + W):
target detected an invalid broadcast address 0b111_1110 + W
1001: TE1 error (CCC code):
target detected a parity error on a CCC code via a parity check (vs. T bit)
1010: TE2 error (write data):
target detected a parity error on a write data via a parity check (vs. T bit)
1011: TE3 error (assigned address during dynamic address arbitration):
target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
1101: TE5 error (transaction after detecting CCC):
target detected an illegally formatted CCC
1110: TE6 error (monitoring error):
target detected that transmitted data on the bus is different from expected
others: reserved

52.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							Res.
								r	r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
r	r	r	r	r	r	r	r						r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **RADD[6:0]**: Received target address (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RCODE[7:0]**: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **IBIRDCNT[2:0]**: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

52.16.13 I3C event register (I3C_EVR)

Address offset: 0x050

Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPF	DEFF	INT UPDF	AS UPDF	RSTF	MRL UPDF	MWL UPDF	DA UPDF	STAF	GETF	WKPF	Res.	HJF	CR UPDF	CRF	IBI ENDF
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIF	Res.	Res.	Res.	ERRF	RXTGT ENDF	FCF	Res.	RX LASTF	TX LASTF	RX FNEF	TX FNFF	SFNEF	CFNFF	TXFEF	CFEF
r				r	r	r		r	r	r	r	r	r	r	r

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.

Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop).

Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.

Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.

The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.

The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.

The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).

The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.

The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that an error occurred.Then, software must read I3C_SER to get the error type.

The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.

Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.

Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

Note: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty.

This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target.

This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

Note: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

52.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
rw				rw	rw	rw				rw	rw	rw	rw		

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bits 1:0 Reserved, must be kept at reset value.

52.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CCRUPDF	CCRF	CIBIENDF
w	w	w	w	w	w	w	w	w	w	w		w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w				w	w	w									

Bit 31 **CGRPF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF

Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF

Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

52.16.16 I3C masked interrupt status register (I3C_MISR)

Address offset: 0x05C

Reset value: 0x0000 0000

This is a read register, containing the masked interrupt status for each individual event flag bit. Each bit is the logical AND of the corresponding interrupt enable bit of the I3C_IER register and the event flag bit of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPMIS**: DEFGRPA CCC masked interrupt status (when the I3C acts as target)

Bit 30 **DEFMIS**: DEFTGTS CCC masked interrupt status (when the I3C acts as target)

Bit 29 **INTUPDMIS**: ENEC/DISEC CCC masked interrupt status (when the I3C acts as target)

Bit 28 **ASUPDMIS**: ENTASx CCC masked interrupt status (when the I3C acts as target)

Bit 27 **RSTMIS**: reset pattern masked interrupt status (when the I3C acts as target)

Bit 26 **MRLUPDMIS**: SETMRL CCC masked interrupt status (when the I3C acts as target)

Bit 25 **MWLUPDMIS**: SETMWL CCC masked interrupt status (when the I3C acts as target)

Bit 24 **DAUPDMIS**: ENTDAA/RSTDAA/SETNEWDA CCC masked interrupt status (when the I3C acts as target)

Bit 23 **STAMIS**: format 1 GETSTATUS CCC masked interrupt status (when the I3C acts as target)

Bit 22 **GETMIS**: GETxxx CCC masked interrupt status, except GETSTATUS of format 1 (when the I3C acts as target)

Bit 21 **WKPMIS**: Wake-up/missed start masked interrupt status (when the I3C acts as target)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJMIS**: Hot-join masked interrupt status (when the I3C acts as controller)

Bit 18 **CRUPDMIS**: Controller-role update masked interrupt status (when the I3C acts as target)

Bit 17 **CRMIS**: Controller-role request masked interrupt status (when the I3C acts as controller)

Bit 16 **IBIENDMIS**: IBI end masked interrupt status (when the I3C acts as target)

Bit 15 **IBIMIS**: IBI request masked interrupt status (when the I3C acts as controller)

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRMIS**: error masked interrupt status (whatever the I3C acts as controller/target)

Bit 10 **RXTGTENDMIS**: target-initiated read end masked interrupt status (when the I3C acts as controller)

Bit 9 **FCMIS**: frame complete masked interrupt status (whatever the I3C acts as controller/target)

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEMIS**: RX-FIFO not empty masked interrupt status (whatever the I3C acts as controller/target)

Bit 4 **TXFNFMIS**: TX-FIFO not full masked interrupt status (whatever the I3C acts as controller/target)

Bit 3 **SFNEMIS**: S-FIFO not empty masked interrupt status (when the I3C acts as controller)

Bit 2 **CFNFMIS**: C-FIFO not full masked interrupt status (when the I3C acts as controller)

Bits 1:0 Reserved, must be kept at reset value.

52.16.17 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT[1:0]		AS[1:0]		HJEN	Res.	CREN	IBIEN
							r	r	r	r	r	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)

This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid.

This bit is cleared by hardware when the target receives a frame start.

When the device is not in Stop mode:

– If RSTVAL = 1: when RSTF in the I3C_EVR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.

– If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the I3C peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the I3C peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset:

– the software writes and sets the SYSRESETREQ control bit of the AIRCR register, when the device is controlled by a Cortex®-M.

11: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled

Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: IBI request disabled

1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address

When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.

When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)

When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.

When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

52.16.18 I3C device x characteristics register (I3C_DEVRx)

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK
r												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	

Bit 31 **DIS**: Disables writes to DA[6:0] (when the I3C acts as controller)

When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS= 1) to lock the configured DA[6:0] and IBIDEN values.

Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS= 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.

0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed

1: write to DA[7:0] and to IBIDEN is disabled/locked

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted

1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18 **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x

1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17 **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.

Bit 16 **IBIACK**: IBI request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x:

0: an IBI request from target x must be NACK-ed

- After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on)

1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.
- After the ACK, the controller logs the IBI payload data, if any, depending on I3C_DEVRx.IBIDEN.
- The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled;
- Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: Assigned I3C dynamic address to target x (when the I3C acts as controller)

When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x.

Writing to this field has no impact when the read field DIS = 1 in the I3C_DEVRx register.

Bit 0 Reserved, must be kept at reset value.

52.16.19 I3C maximum read length register (I3C_MAXRLR)

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **IBIP[2:0]**: IBI payload data maximum size, in bytes (when I3C acts as target)
This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged.This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)
001: 1 byte (mandatory data byte MDB[7:0]
010: 2 bytes (including first MDB[7:0])
011: 3 bytes (including first MDB[7:0])
100: 4 bytes (including first MDB[7:0])
others: same as 100

Bits 15:0 **MRL[15:0]**: Maximum data read length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.

52.16.20 I3C maximum write length register (I3C_MAXWLR)

Address offset: 0x094

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MWL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MWL[15:0]**: Maximum data write length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.
Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.

52.16.21 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLH_I2C[7:0]								SCLL_OD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH_I3C[7:0]								SCLL_PP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I^2C messages, in number of kernel clocks cycles:

$t_{SCLH_I2C} = (SCLH_I2C + 1) \times t_{I3CCLK}$

Note: SCLH_I2C is used to generate t_{DIG_H} (I^2C) timing when communicating with I^2C devices.

Note: With I^2C Fm+ device t_{DIG_Hmin} = 260 ns, with I^2C Fm+ device t_{DIG_Hmin} = 600 ns.

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I^2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

$t_{SCLL_OD} = (SCLL_OD + 1) \times t_{I3CCLK}$

Note: SCLL_OD is used to generate both t_{DIG_L} (I^2C) and $t_{DIG_OD_L}$(I3C) timings.

Note: With I^2C Fm+ device t_{DIG_Lmin} = 500 ns, with I^2C Fm device t_{DIG_Lmin} = 1320 ns.

Note: I3C messages: $t_{DIG_OD_Lmin}$ = 200 ns.

Note: If a single I3C frame is gathering I^2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I^2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

$t_{SCLH_I3C} = (SCLH_I3C + 1) \times t_{I3CCLK}$

Note: SCLH_I3C is used to generate both t_{DIG_H} (I3C) and $t_{DIG_H_MIXED}$ timings.

Note: For mixed bus (with at least one I^2C target): $t_{DIG_H_MIXEDmin}$ = 32 ns and $t_{DIG_H_MIXEDmax}$ = 45 ns (due to I^2C 50 ns spike filter).

Note: For pure I3C bus (with no I^2C targets): t_{DIG_Hmin} = 32 ns.

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

$t_{SCLL_PP} = (SCLL_PP + 1) \times t_{I3CCLK}$

Note: SCLL_PP is used to generate t_{DIG_L} (I3C in PP) timing.

Note: t_{DIG_Lmin} = 32 ns (max 40/60 duty cycle at 12.5 MHz).

52.16.22 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SDA_HD[1:0]		Res.	Res.	Res.	Res.	Res.	FREE[6:0]						
		rw	rw						rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **SDA_HD[1:0]**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t_{HD_PP}):

SDA hold time = (SDA_HD[1:0] + 0.5) x t_{I3CCLK}

Note: when controller: t_{HD_PPmiN} = min (t_{CR} , t_{CF}) + 3 ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t_{CAS} and (I^2C) t_{BUF}. These timings are defined by: t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD[1:0])] x t_{I3CCLK}

Note: For pure I3C bus: t_{CASmin} = 38.4 ns, and t_{CASmax} = 1 µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with I^2C Fm+ device t_{BUFmin} = 0.5 µs, for mixed bus with I^2C Fm device t_{BUFmin} = 1.3 µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to. MIPI timing t_{CASr}. This timing is defined by: t_{CASr} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD[1:0])] x t_{I3CCLK}.

Note: $t_{CASr,\ miN}$ = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by: t_{CBP} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: t_{CBPmiN} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by: t_{CBSr} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: $t_{CBSr,\ miN}$ = 19.2 ns.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0].

This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This time unit is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

When the I3C acts as target:

1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t_{AVAL}). Refer to MIPI timing t_{AVAL} = 1 µs. This timing is defined by: t_{AVAL} = (AVAL[7:0] + 2) x t_{I3CCLK}
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t_{IDLE} = 200 µs . This timing is defined by: t_{IDLE} = (AVAL[7:0] + 2) x 200 x t_{I3CCLK}

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:

1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing $t_{STALLDAA}$ = 15 ms. This timing is defined by: $t_{STALLDAAmax}$ = (AVAL[7:0] + 1) x 15000 x t_{I3CCLK}
2. on ACK/NACK phase of I3C/I^2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t_{STALL} = 100 µs. This timing is defined by: $t_{STALLmax}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for $t_{NEWCRLock}$ before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t_{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same $t_{NEWCRLock}$ time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ANSCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x t_{I3CCLK}
   - ASNCR[1:0] = 01: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}
   - ASNCR[1:0] = 10: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 2000 x t_{I3CCLK}
   - ASNCR[1:0] = 11: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 50000 x t_{I3CCLK}

52.16.23 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I^2C target(s).

This register can be written only when the I3C acts as controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STALL[7:0]								Res.	STALLL	STALLS	STALLR	STALLA	STALLC	STALLD	STALLT
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

t_{SCLL_STALL} = STALL x t_{I3CCLK}

Bit 7 Reserved, must be kept at reset value.

Bit 6 **STALLL:** Controller clock stall enable in the address ACK/NACK phase (before the ninth bit) of a legacy I2C read/write message. This allows the I2C target to prepare the next data byte to send/receive.

0: no stall

1: stall enabled

Bit 5 **STALLS:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C write message. This allows the I2C target to prepare the next data byte to receive.

0: no stall

1: stall enabled

Bit 4 **STALLR:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C read message. This allows the I2C target to prepare the next data byte to send.

0: no stall

1: stall enabled

Bit 3 **STALLA:** Controller clock stall enable on ACK phase

The SCL is stalled (during t_{SCLL_STALL}as defined by STALL) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2 **STALLC:** Controller clock stall enable on PAR phase of CCC

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 1 **STALLD:** Controller clock stall enable on PAR phase of Data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 0 **STALLT:** Controller clock stall enable on T-bit phase of data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

52.16.24 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
									rw				rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable

0: I3C target (no controller capable)

1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.

52.16.25 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DCR[7:0]**: device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device
Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.

52.16.26 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification

1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.

52.16.27 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:

- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRLCAP2[7:0] on the I3C bus is then as follows:

- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CAP GRP	Res.	Res.	Res.	Res.	Res.	CAP DHOFF	Res.	Res.	Res.
						rw						rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CAPGRP**: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities

1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **CAPDHOFF**: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off

1: this I3C needs additional time to process a controller-role hand-off

Bits 2:0 Reserved, must be kept at reset value.

52.16.28 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time t_{SCO} ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (t_{SCO} > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]							
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS[1:0]	
						rw	rw							rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCO**: clock-to-data turnaround time (t_{SCO})

This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time t_{SCO} (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.

0: $t_{SCO} \leq 12$ ns

1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16 **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)

This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FMT[1:0]**: GETMXDS CCC format

This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).

This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.

00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)

01: format 2: (5 bytes w.ith MaxWr with no defining byte, MaxRd, MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.

10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs

11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **HOFFAS[1:0]**: Controller hand-off activity state

This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.

This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.

00: activity state 0 is the initial activity state of this I3C before and when becoming controller

01: activity state 1 is the initial activity state of this I3C when becoming controller

10: activity state 2 is the initial activity state of this I3C when becoming controller

11: activity state 3 is the initial activity state of this I3C when becoming controller

52.16.29 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

The 48-bit provisioned ID on the I3C bus is then returned as:

- provisioned ID [47:32] = 0x0208
- provisioned ID [31:16] = 0x1381
- provisioned ID [15:0] = {MIPIID[3:0], 0x000}

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIPIMID[14:0]															IDTSEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:17 **MIPIMID[14:0]**: 15-bit MIPI manufacturer ID
This read field is the 15-bit STMicroelectronics MIPI ID (0x0104).
This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 **IDTSEL**: provisioned ID type selector
This field is set as 0 (vendor fixed value).
This field represents bit[32] of the 48-bit provisioned ID.
Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 **MIPIID[3:0]**: 4-bit MIPI Instance ID
This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus.
This field represents bits[15:12] of the 48-bit provisioned ID.
Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

52.16.30 I3C register map

Table 612. I3C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	I3C_CR	MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0] / CCC[7:0]							RNW	DCNT[15:0]																
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	I3C_CFGR	Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCFDIS	CFLUSH	CDMAEN	TMODE	SMODE	SFLUSH	SDMAEN	Res.	TXTHRES	TXFLUSH	TXDMAEN	Res.	RXTHRES	RXFLUSH	RXDMAEN	HJACK	Res.	HKSDAEN	EXITPTRN	RSTPTRN	NOARBH	CRINIT	EN
	Reset value		0								0	0	0	0	0	0	0		0	0	0		0	0	0	0		0	0	0	0	0	0
0x010	I3C_RDR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	I3C_RDWR	RDB3[7:0]								RDB2[7:0]								RDB1[7:0]								RDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	I3C_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	I3C_TDWR	TDB3[7:0]								TDB2[7:0]								TDB1[7:0]								TDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	I3C_IBIDR	IBIDB3[7:0]								IBIDB2[7:0]								IBIDB1[7:0]								IBIDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	I3C_TGTTDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	TGTTDCNT[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	I3C_SR	MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.	XDCNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0						0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	I3C_SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x040	I3C_RMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							s e T R	RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDC NT[2:0]		
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0	0	0	0						0	0	0

Table 612. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	I3C_EVR	GRPF	DEFF	INTUPDF	ASUPDF	RSTF	MRLUPDF	MWLUPDF	DAUPDF	STAF	GETF	WKPF	Res.	HJF	CRUPDF	CRF	IBIENDF	IBIF	Res.	Res.	Res.	ERRF	RXTGTENDF	FCF	Res.	RXLASTF	TXLASTF	RXFNEF	TXFNFF	SFNEF	CFNFF	TXFEF	CFEF
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0		0	0	0	0	0	0	1	1
0x054	I3C_IER	GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE	IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0				0	0	0	0		
0x058	I3C_CEVR	CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CRRUPDF	CCRF	CIBIENDF	CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0									
0x05C	I3C_MISR	GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS	IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0				0	0	0	0		
0x060	I3C_DEVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT [1:0]		AS [1:0]		HJEN	Res.	CREN	IBIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
	Reset value								0	0	0	0	0	0		0	0									0	0	0	0	0	0	0	0
0x060 + 4 * x, (x = 1 to 4)	I3C_DEVRx	DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
	Reset value	0												0	0	0	0									0	0	0	0	0	0	0	
0x090	I3C_MAXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP [2:0]			MRL15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	I3C_MAXWLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MWL15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	I3C_TIMINGR0	SCLH_I2C[7:0]								SCLL_OD[7:0]								SCLH_I3C[7:0]								SCLL_PP[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	I3C_TIMINGR1	Res.	Res.	SDA_HD		Res.	Res.	Res.	Res.	Res.	FREE[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
	Reset value			0	0						0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A8	I3C_TIMINGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVAL[7:0]								Res.	STALLL	STALLS	STALLR	STALLA	STALLC	STALLD	STALLT
	Reset value																	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0

Table 612. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C0	I3C_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
	Reset value																										0				0		0
0x0C4	I3C_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C8	I3C_GETCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0														
0x0CC	I3C_CRCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPGR	Res.	Res.	Res.	Res.	Res.	CAPDHOFF	Res.	Res.	Res.
	Reset value																							0						0			
0x0D0	I3C_GETMXDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS [1:0]	
	Reset value								0	0	0	0	0	0	0	0	0							0	0							0	0
0x0D4	I3C_EPIDR	MIPIMID[14:0]															IDTSEL	MIPIID[3:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0												

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

53 Universal synchronous/asynchronous receiver transmitter (USART/UART)

53.1 USART introduction

The USART offers a flexible means to perform full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

53.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit

  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from low-mode mode

53.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability (compatible with all LIN specification versions; complete LIN support requires additional software management).
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

53.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 613. Instance implementation on STM32G56x/55x/54x/53x

Instance	STM32G53x/54x	STM32G55x/56x
USART1	Full	Full
USART2	Full	Full
USART3	-	Full
UART4	Basic	Basic
UART5	Basic	Basic
UART6	Basic	Basic
LPUART1	Low-power	Low-power

Table 614. USART/LPUART features

Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X

Table 614. USART/LPUART features (continued)

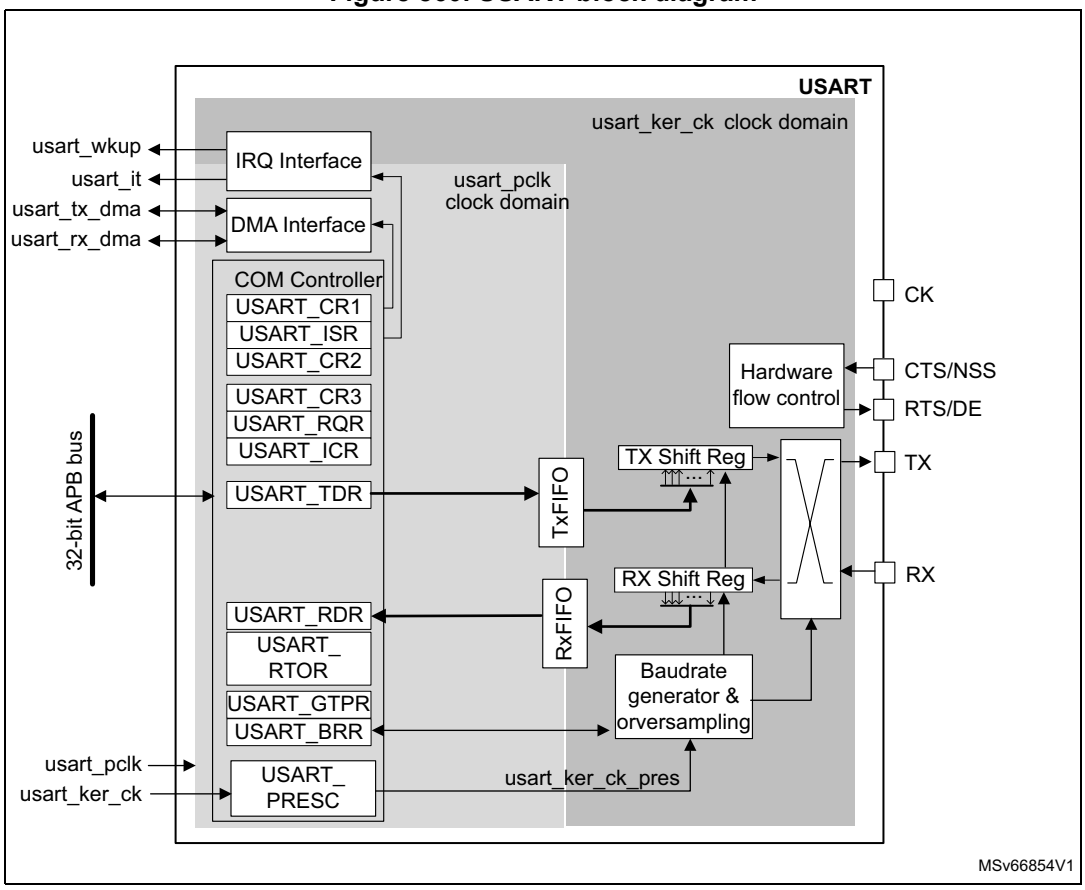
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop mode.

53.5 USART functional description

53.5.1 USART block diagram

Figure 860. USART block diagram



53.5.2 USART pins and internal signals

Description of USART input/output pins

- USART bidirectional communications

  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input)

    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

  – **TX** (Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

- RS232 hardware flow control mode

  The following pins are required in RS232 hardware flow control mode:

  – **CTS** (Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request To Send)

    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

- Synchronous SPI master/slave mode and smartcard mode

  The following pins are required in synchronous master/slave mode and smartcard mode:

  – **CK**

    This pin acts as clock output in synchronous SPI master and smartcard modes.

    It acts as clock input in synchronous SPI slave mode.

    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.

    In smartcard mode, CK output provides the clock to the smartcard.

  – **NSS**

    This pin acts as Slave Select input in synchronous slave mode.

Refer to *Table 615* and *Table 616* for the list of USART input/output pins and internal signals.

Table 615. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 616. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request

53.5.3 USART clocks

The simplified block diagram given in *Section 53.5.1: USART block diagram* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

53.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 861*):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

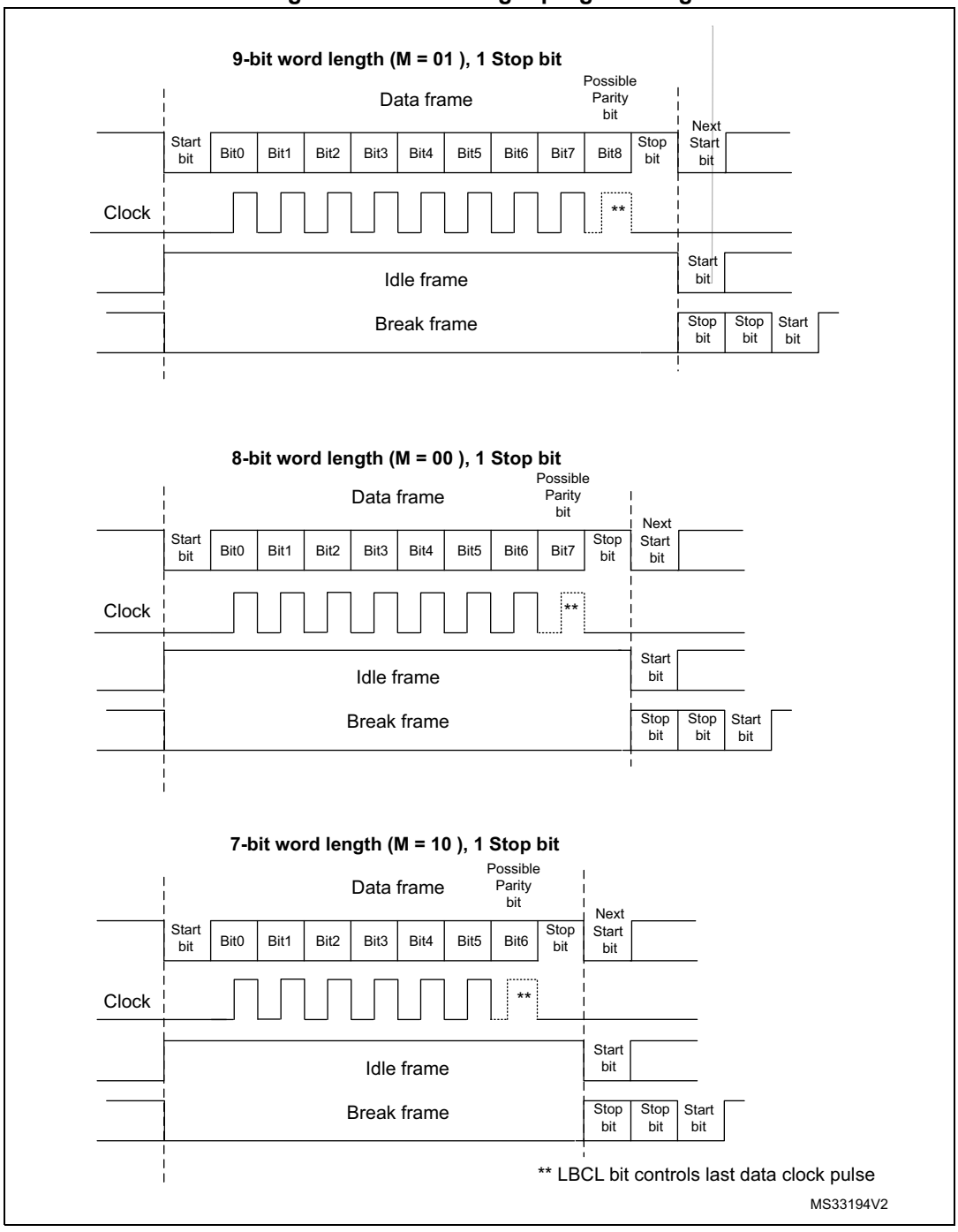
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 861. Word length programming



53.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size – 1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

53.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

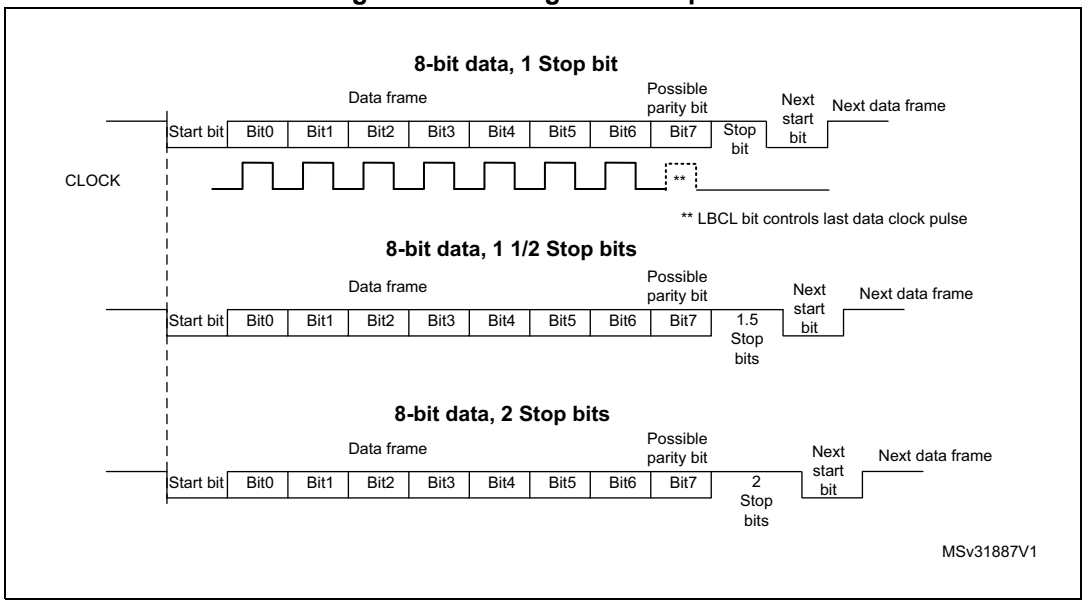
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Section 53.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 862. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 53.5.20: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

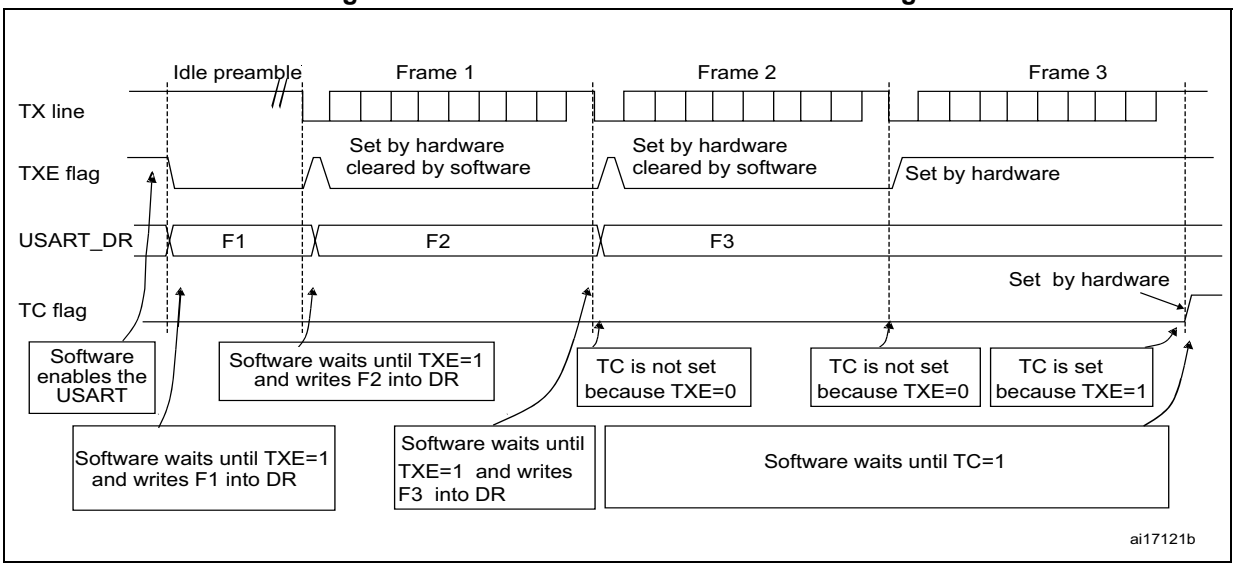
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 863*).

Figure 863. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 861*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

53.5.7 USART receiver

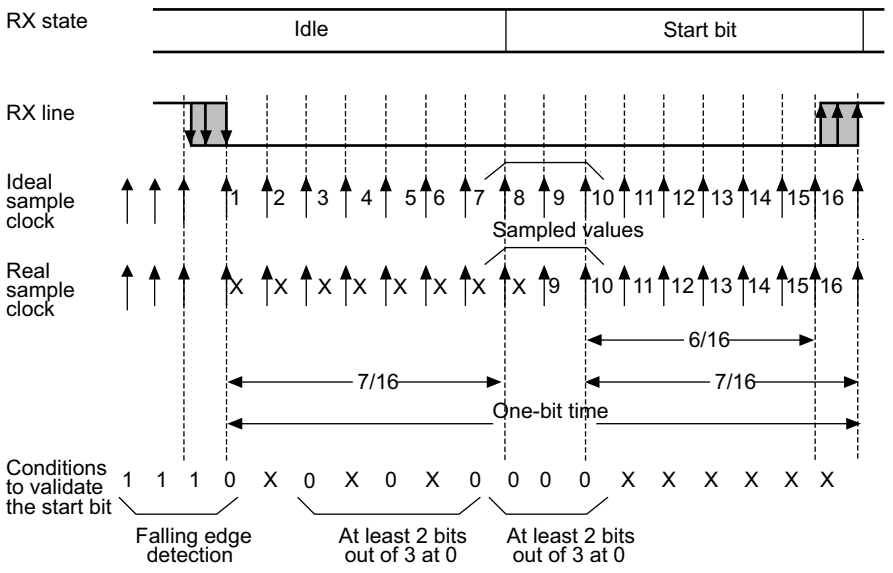
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 864. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 53.5.20: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section : Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

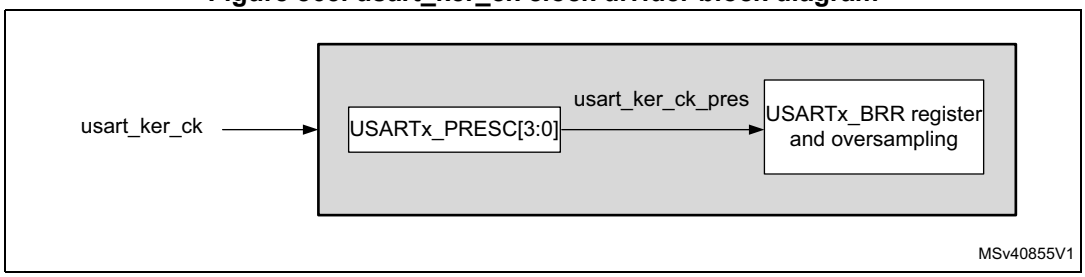
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section : Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 865. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 866* and *Figure 867*).

Depending on the application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 53.5.9: Tolerance of the USART receiver to clock deviation*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on the application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 617*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 53.5.9: Tolerance of the USART receiver to clock deviation*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 866. Data sampling when oversampling by 16

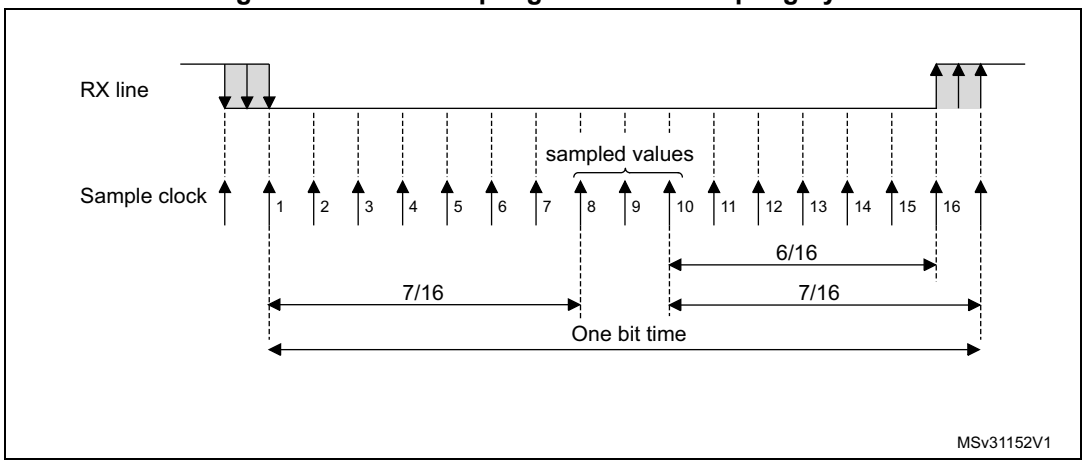


Figure 867. Data sampling when oversampling by 8

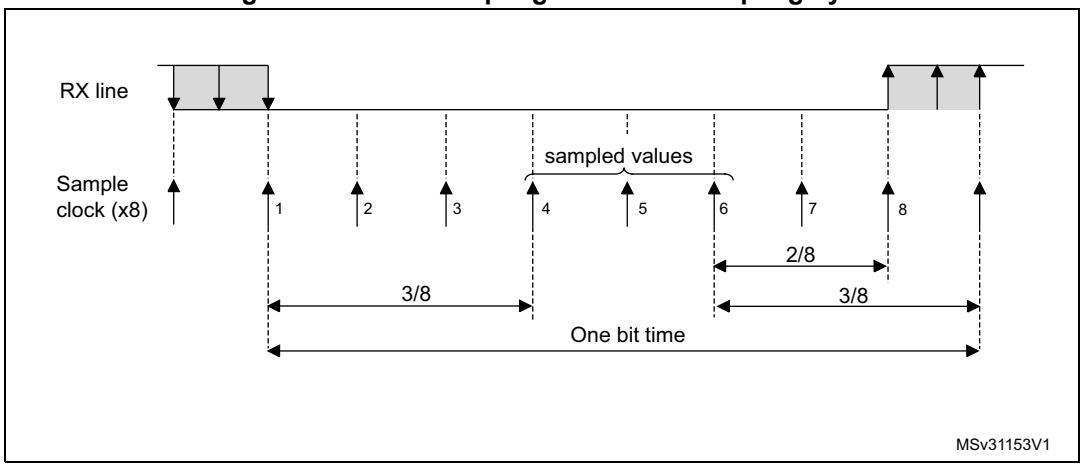


Table 617. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (smartcard mode)**

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 53.5.17: USART receiver timeout* for more details).
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

53.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 bauds with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR[15:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x0683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[15:0] = 0x0681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR[15:0] = USARTDIV = 52 = 0x0034
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR[15:0] = 0x0064

53.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 618*, *Table 619*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 618. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 619. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 618 and Table 619 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

53.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/32763 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

53.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

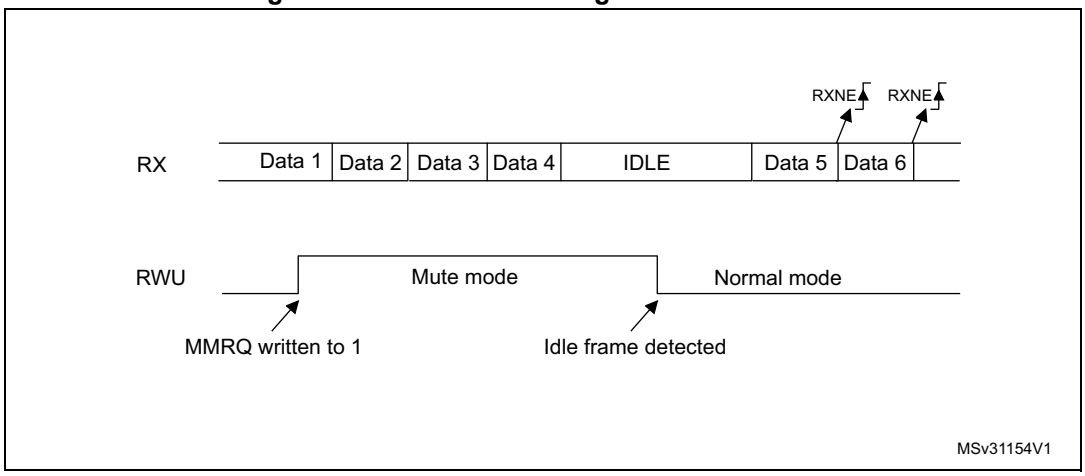
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 868*.

Figure 868. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

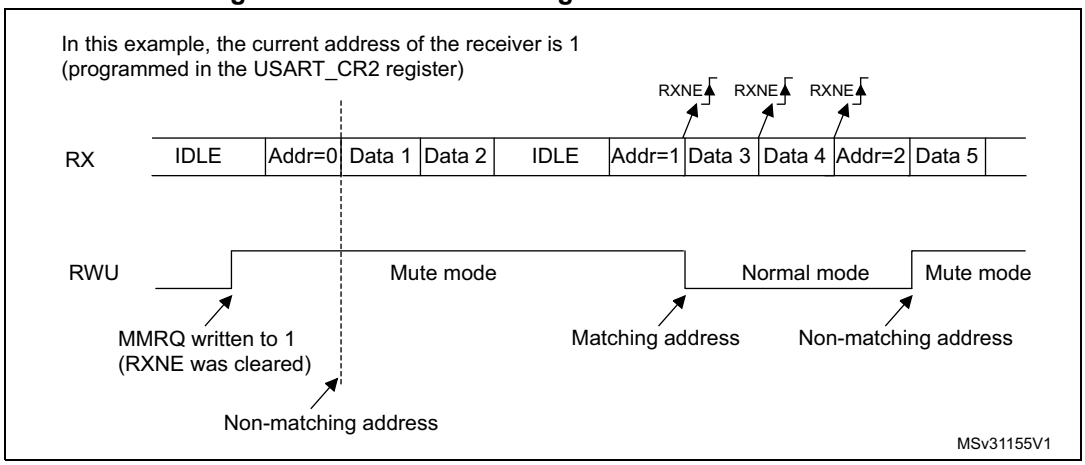
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 869*.

Figure 869. Mute mode using address mark detection



53.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

53.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 620*.

Table 620. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

53.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 53.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 53.5.5* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 870*.

Examples of break frames are given in *Figure 871*.

Figure 870. Break detection in LIN mode (11-bit break length - LBDL bit is set)

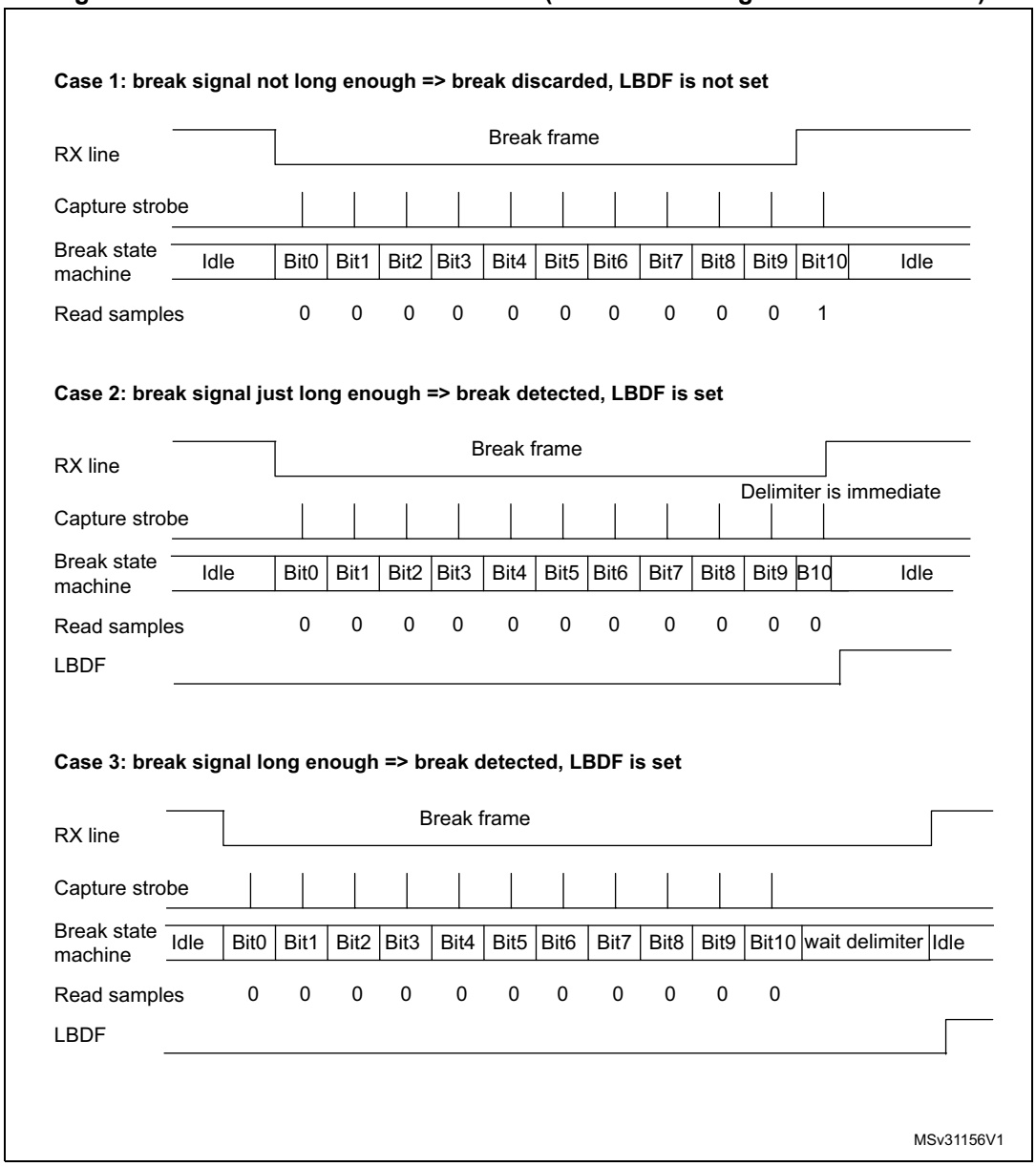
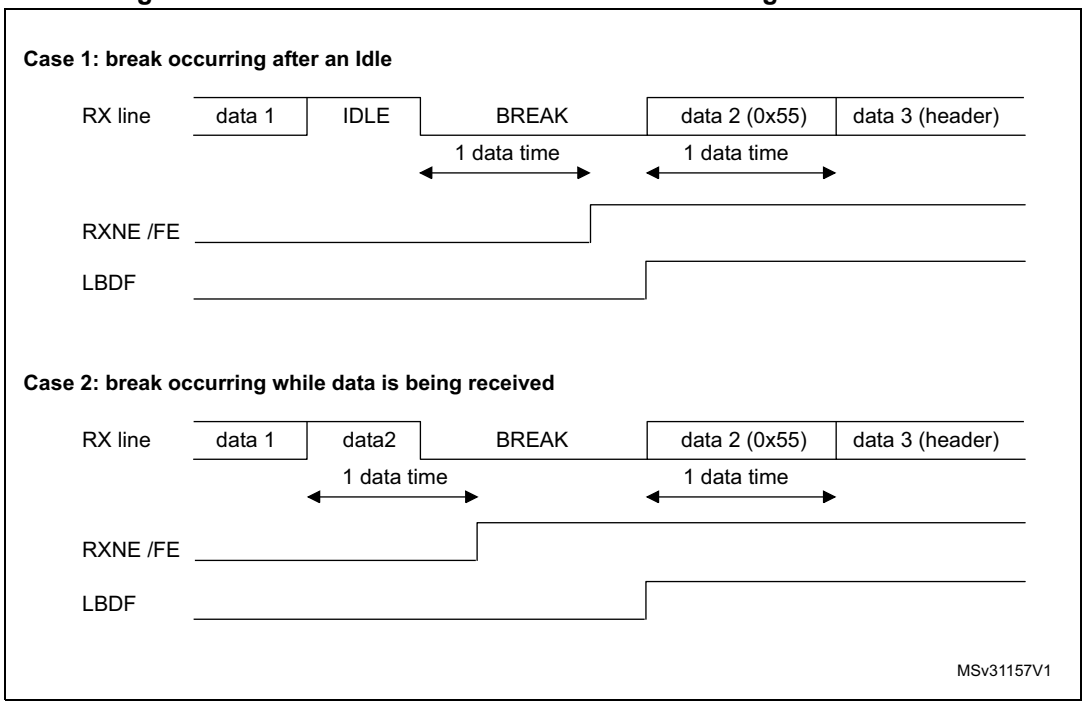


Figure 871. Break detection in LIN mode vs. Framing error detection



53.5.15 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 872*, *Figure 873* and *Figure 874*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 872. USART example of synchronous master transmission

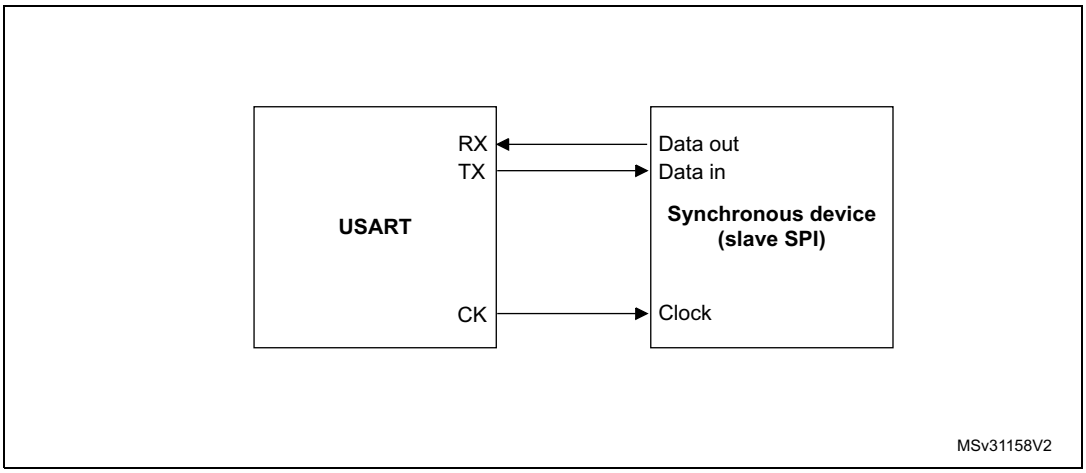


Figure 873. USART data clock timing diagram in synchronous master mode (M bits = 00)

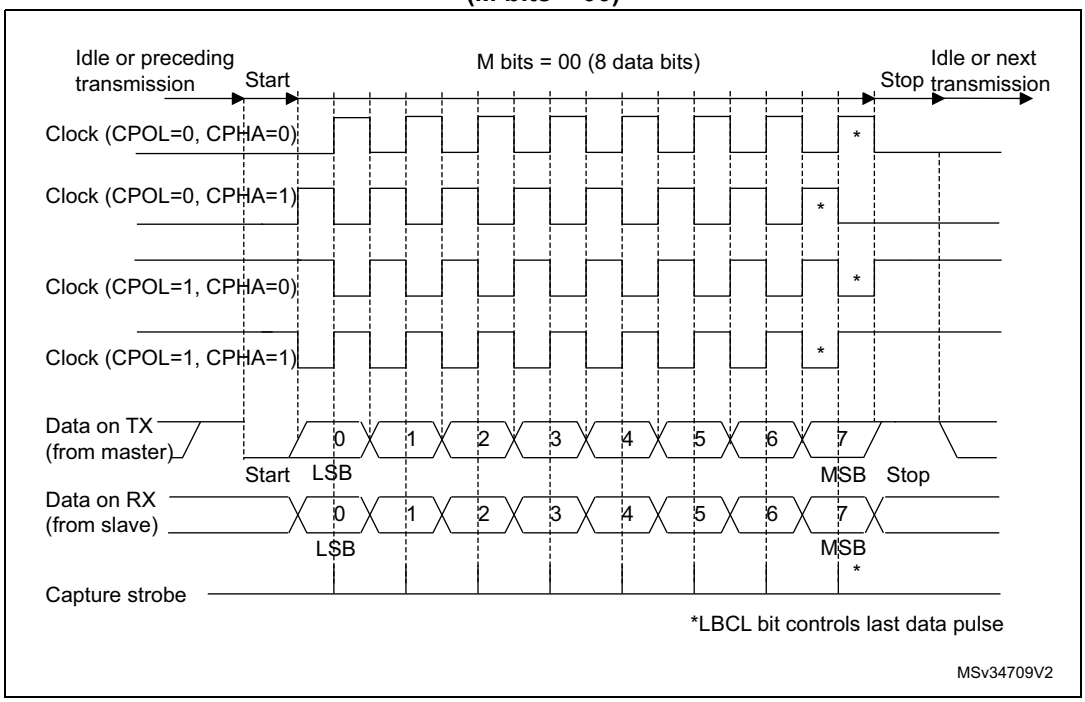
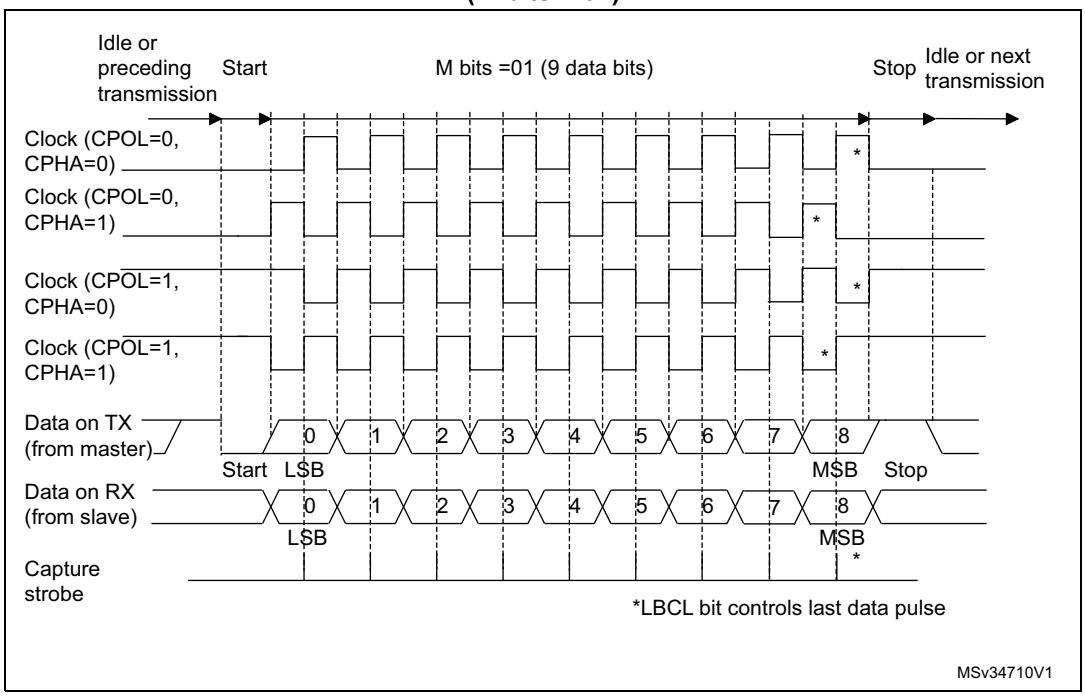


Figure 874. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

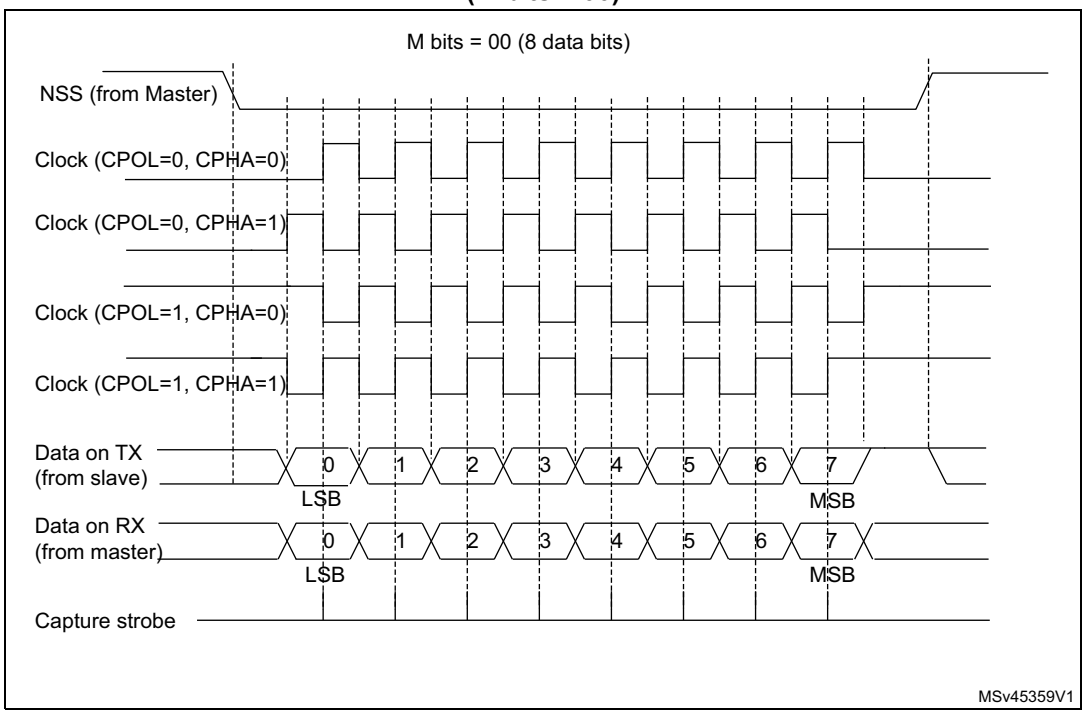
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 875*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 875. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usart_ker_ck *cycles before the first CK edge.*

53.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

53.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

53.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 53.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

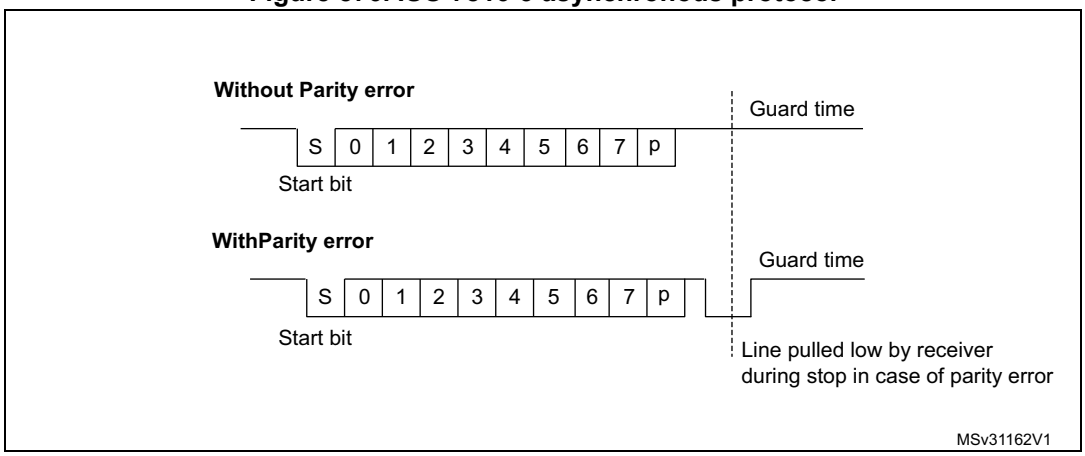
The USART must be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = 11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 876 shows examples of what can be seen on the data line with and without parity error.

Figure 876. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

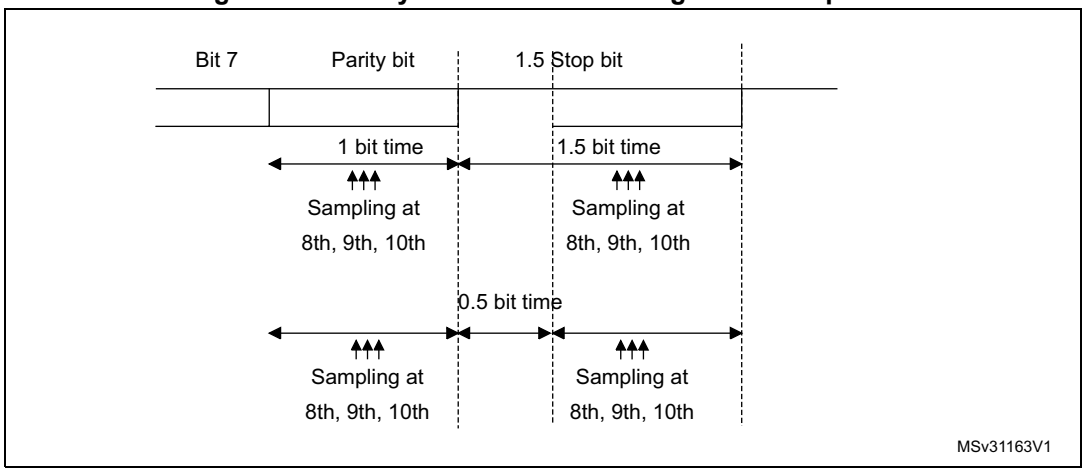
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure the minimum 2-baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 877 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 877. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT – 11 or CWT – 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV =1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

53.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 53.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 878*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

The SIR transmit encoder modulates the non return-to-zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is

ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 879*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to 1 stop bit.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3 / 16 of the bit period. Instead, the width of the pulse is three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 878. IrDA SIR ENDEC block diagram

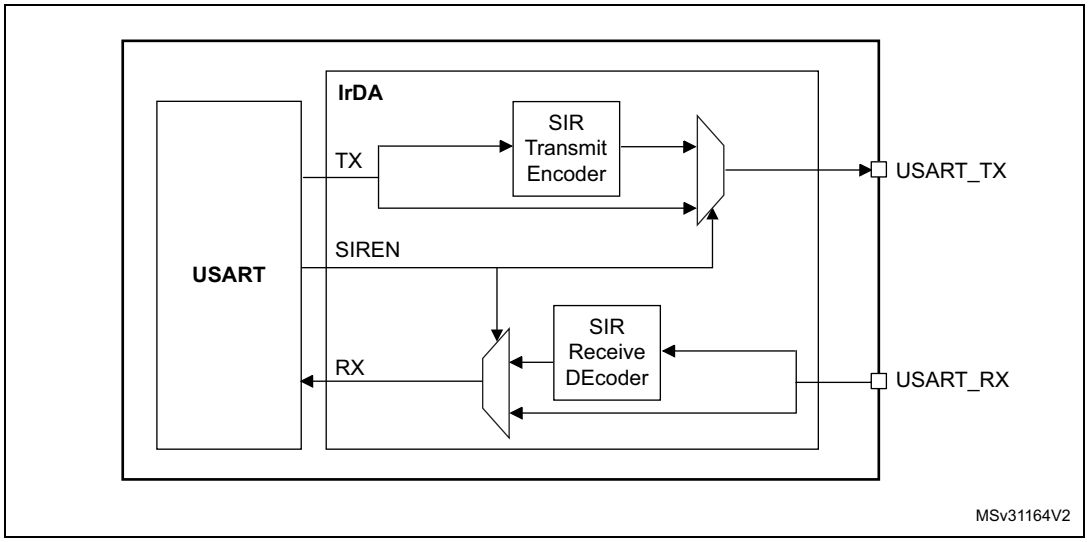
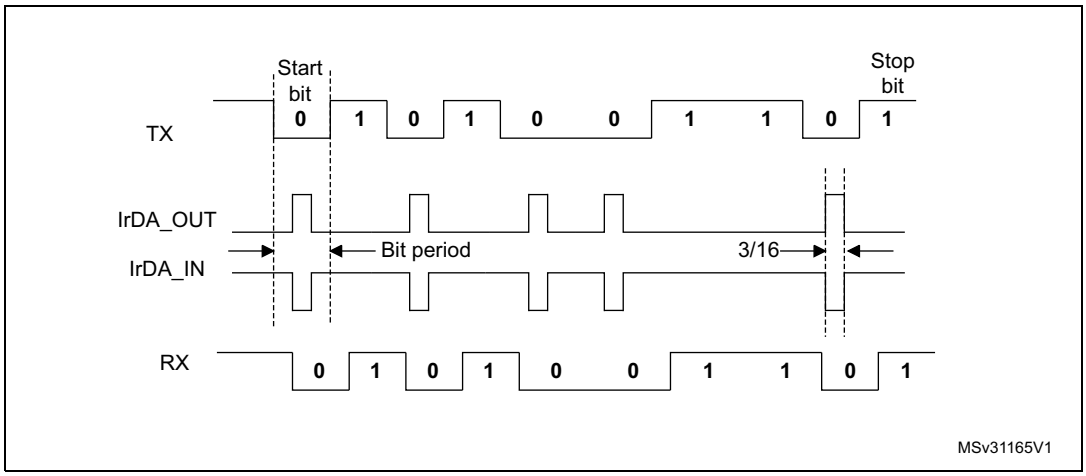


Figure 879. IrDA data modulation (3/16) - normal mode



53.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 53.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 53.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

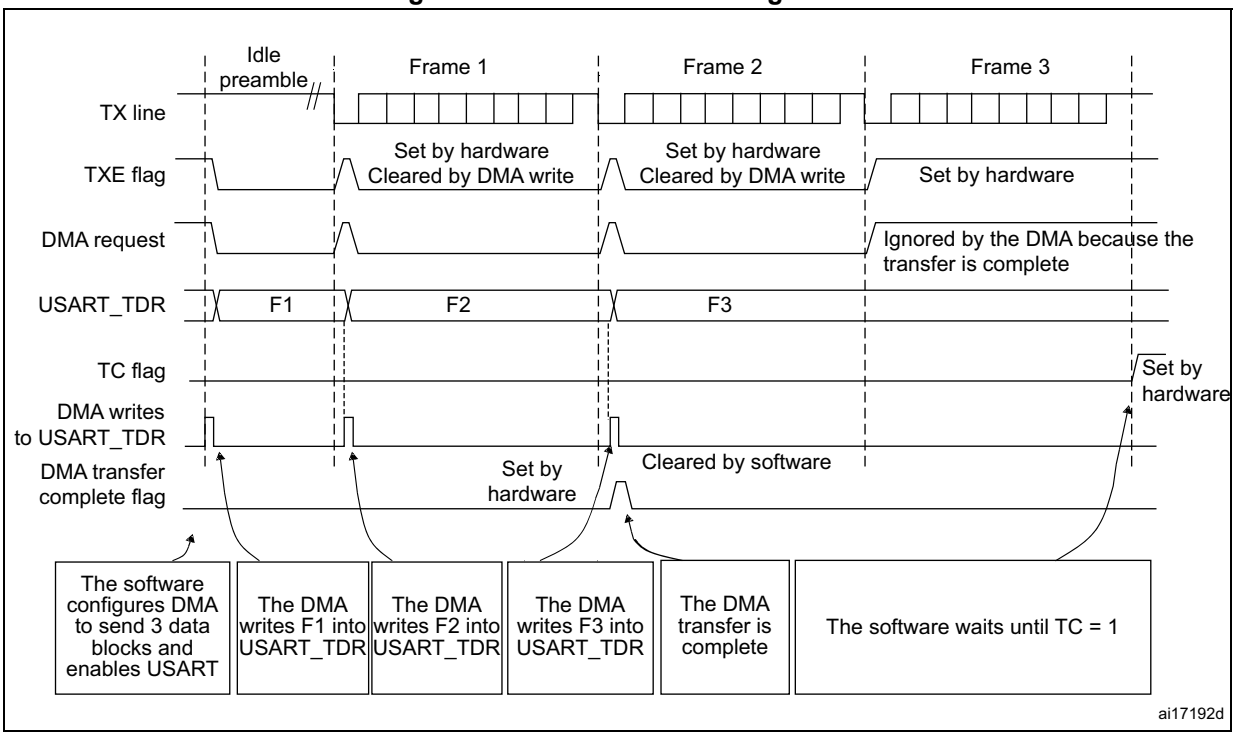
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 880. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

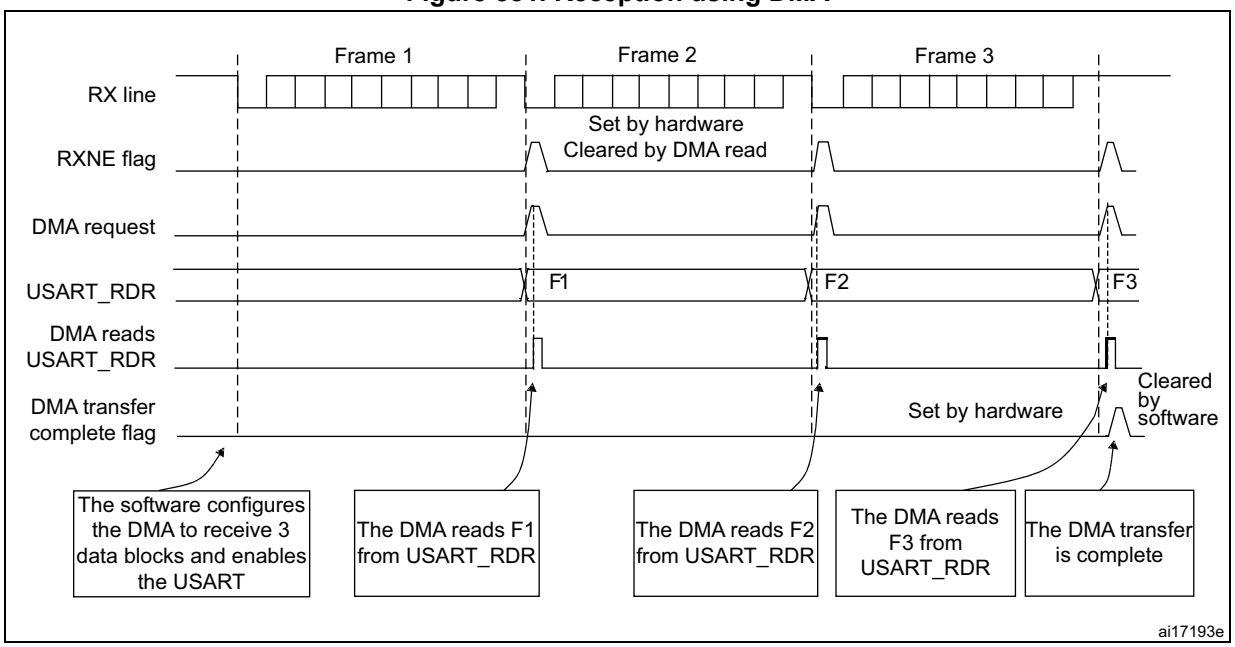
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 881. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

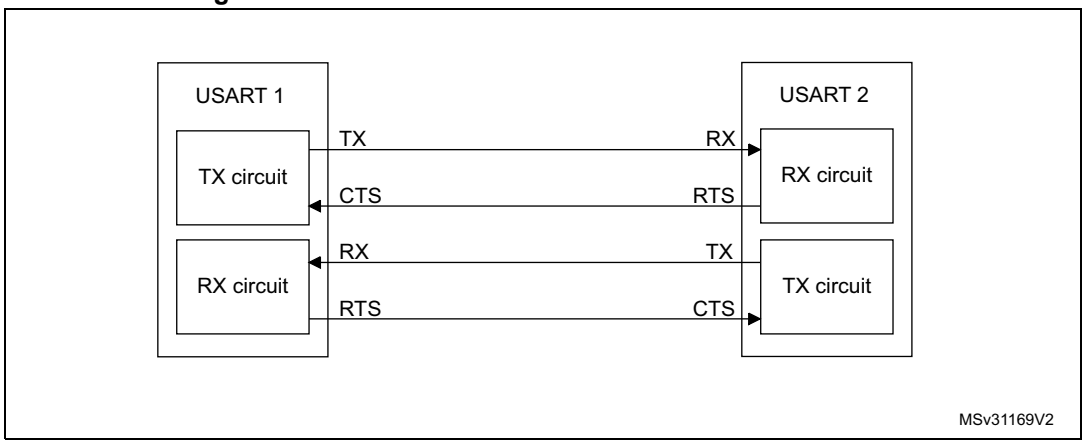
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

53.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 882* shows how to connect two devices in this mode:

Figure 882. Hardware flow control between two USARTs

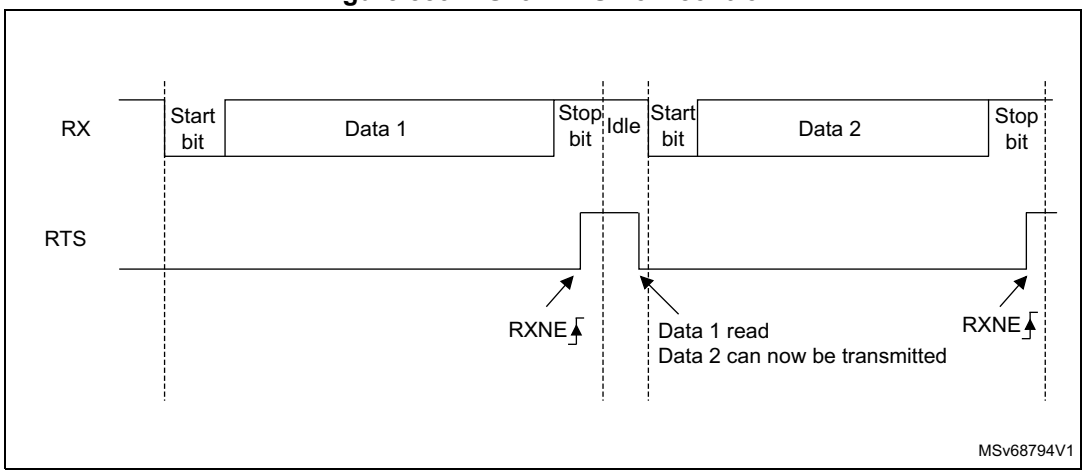


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 883* shows an example of communication with RTS flow control enabled.

Figure 883. RS232 RTS flow control



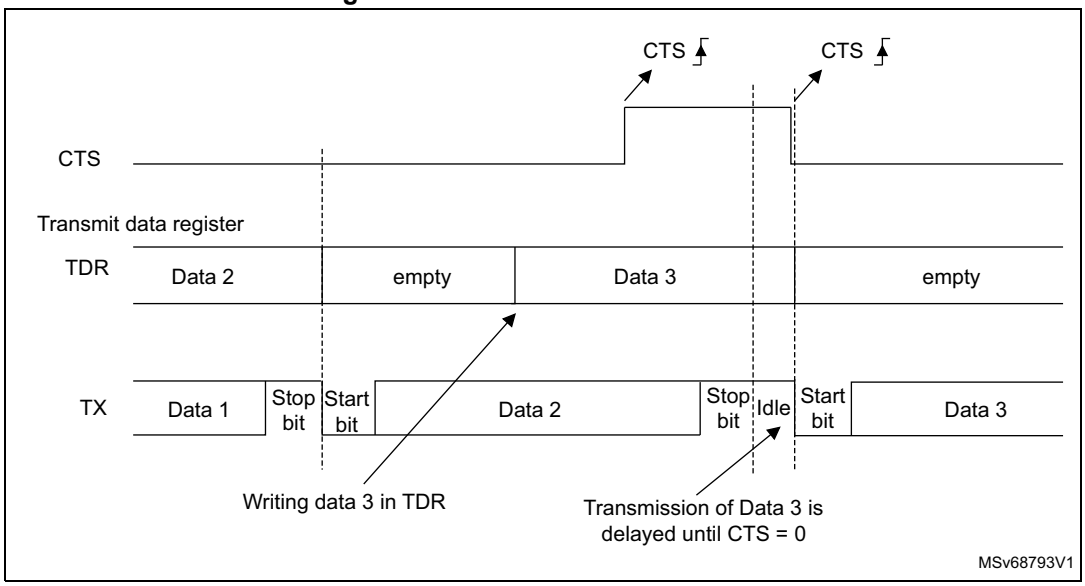
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 884* shows an example of communication with CTS flow control enabled.

Figure 884. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (driver enable) signal. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

53.5.22 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 885 shows the USART behavior when the wake-up event is verified.

Figure 885. Wake-up event verified (wake-up event = address match, FIFO disabled)

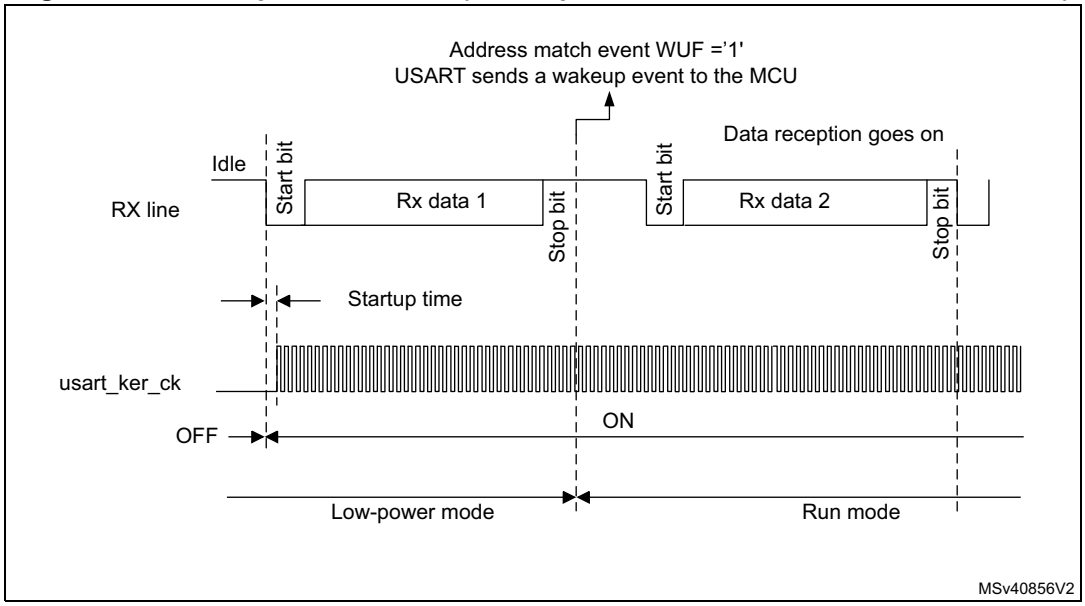
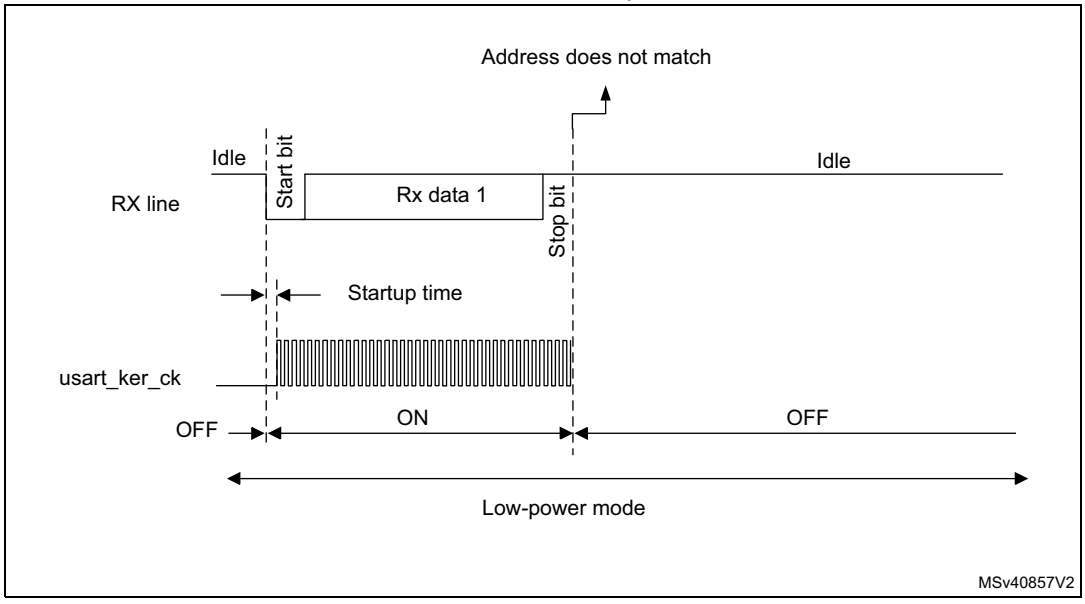


Figure 886 shows the USART behavior when the wake-up event is not verified.

Figure 886. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as a wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 53.5.9: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 618: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = USART receiver tolerance – DREC = 3.41% – 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

53.6 USART in low-power modes

Table 621. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 53.4: USART implementation* to know if the wake-up from low-mode mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

53.7 USART interrupts

Refer to *Table 622* for a detailed description of all USART interrupt requests.

Table 622. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR(2)		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission complete before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 622. USART interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO not empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO full	RXFF(3)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		No	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from low-mode mode feature. Refer to *Section 53.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
4. When OVRDIS = 0.

53.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

53.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method
This bit determines the USART wake-up method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

53.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits

M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits

This bit can only be written when the USART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXE = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable

This bit is set and cleared by software.
0: Interrupt is inhibited
1: USART interrupt is generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.
In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.
When this bit is set, the USART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

53.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter is equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter is equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: RX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 866* and *Figure 867*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

53.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF = 1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.
If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE = 0).
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the USART is disabled (UE = 0).
Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation on page 2782.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

53.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever WUF = 1 in the USART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

53.8.6 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]:** USART baud rate

BRR[15:4]

BRR[15:4] correspond to USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

53.8.7 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The transmission complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.

This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA modes are not supported. Refer to Section 53.4: USART implementation.

53.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the block length in smartcard T = 1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1 – LEC / 2 – CRC) – 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In smartcard mode, the block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the receiver timeout feature is not supported. Refer to Section 53.4: USART implementation.

53.8.9 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

53.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.
An interrupt is generated if the TXFTIE bit (bit 31) is set in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 53.4: USART implementation.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit = 1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed, and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by a write to the USART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 53.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

53.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 53.4: USART implementation.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1 in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register empty

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 53.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

53.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

53.8.13 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 53.5.1: USART block diagram*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

53.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 53.5.1: USART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

53.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation.

53.8.16 USART register map

Table 623. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			TCBGTIE	TXFTIE	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value								0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 623. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_ GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_ RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

54 Low-power universal asynchronous receiver transmitter (LPUART)

54.1 LPUART introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

54.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from low-mode mode

54.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 624. Instance implementation on STM32G56x/55x/54x/53x

Instance	STM32G53x/54x	STM32G55x/56x
USART1	Full	Full
USART2	Full	Full
USART3	-	Full
UART4	Basic	Basic
UART5	Basic	Basic
UART6	Basic	Basic
LPUART1	Low-power	Low-power

Table 625. USART/LPUART features

Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X

Table 625. USART/LPUART features (continued)

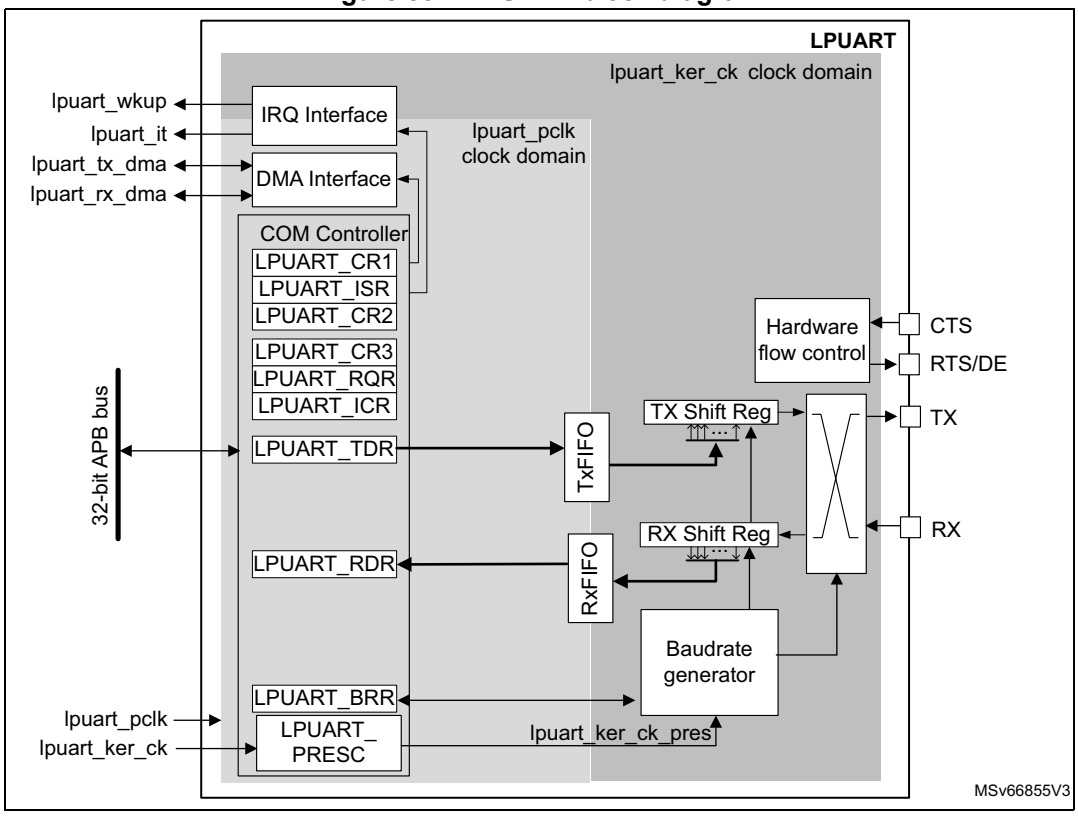
Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X(2)	X(2)	X(2)

1. X = supported.
2. Wake-up supported from Stop mode.

54.4 LPUART functional description

54.4.1 LPUART block diagram

Figure 887. LPUART block diagram



54.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input):

    RX is the serial data input.

  – **TX (**Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

  The RS232 hardware flow control mode requires the following pins:

  – **CTS (**Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request to send)

    When it is low, this signal indicates that the LPUART is ready to receive data.

- RS485 hardware flow control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to *Table 626* and *Table 627* for the list of LPUART input/output pins and internal signals.

Table 626. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE(1)	Output	Driver enable

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 627. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt

Table 627. LPUART internal input/output signals (continued)

Signal name	Signal type	Description
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request

54.4.3 LPUART clocks

The simplified block diagram given in *Section 54.4.1: LPUART block diagram* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

54.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 861*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

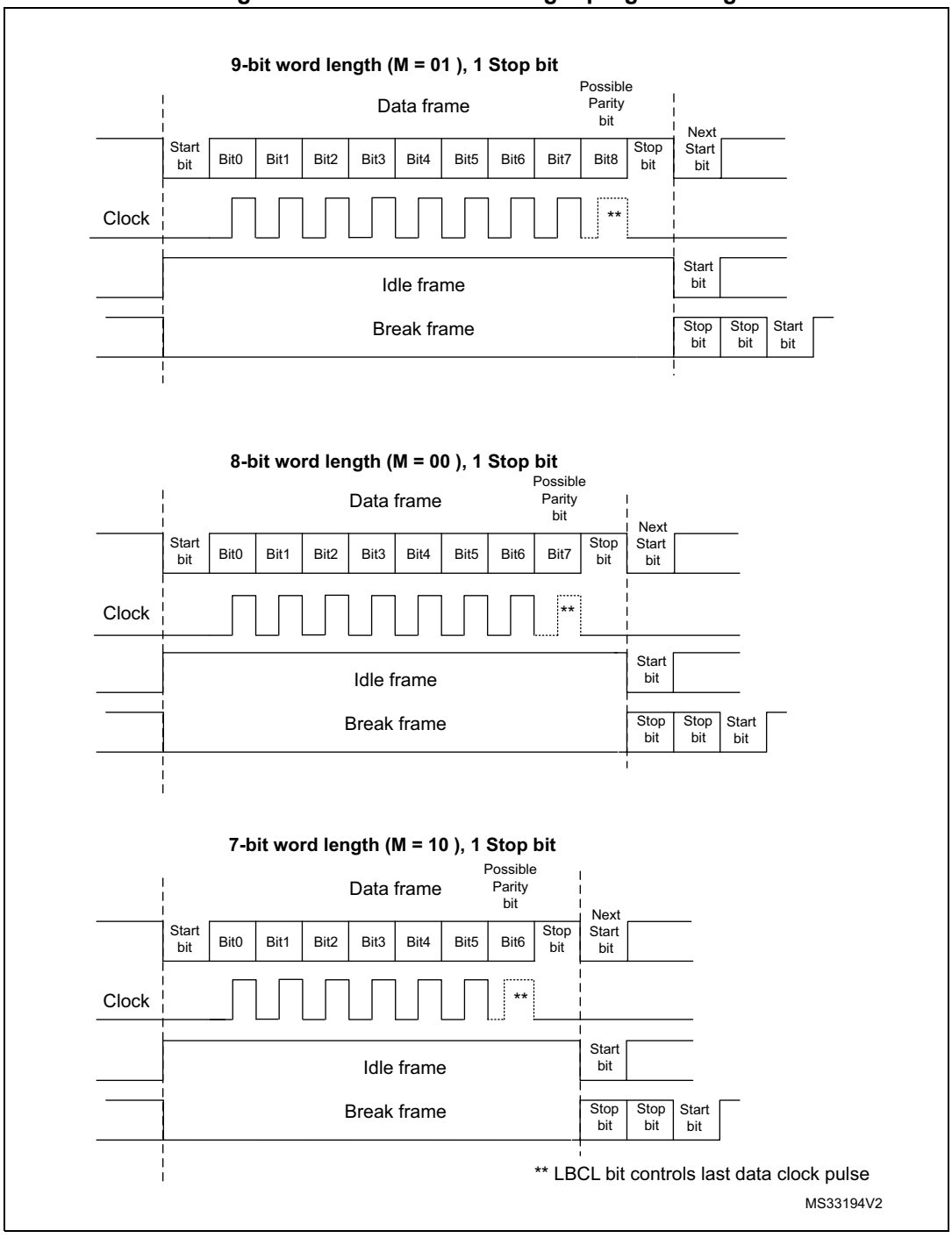
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 888. LPUART word length programming



54.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size – 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

54.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The transmit enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Section 54.4.1: LPUART block diagram*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit bit, which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

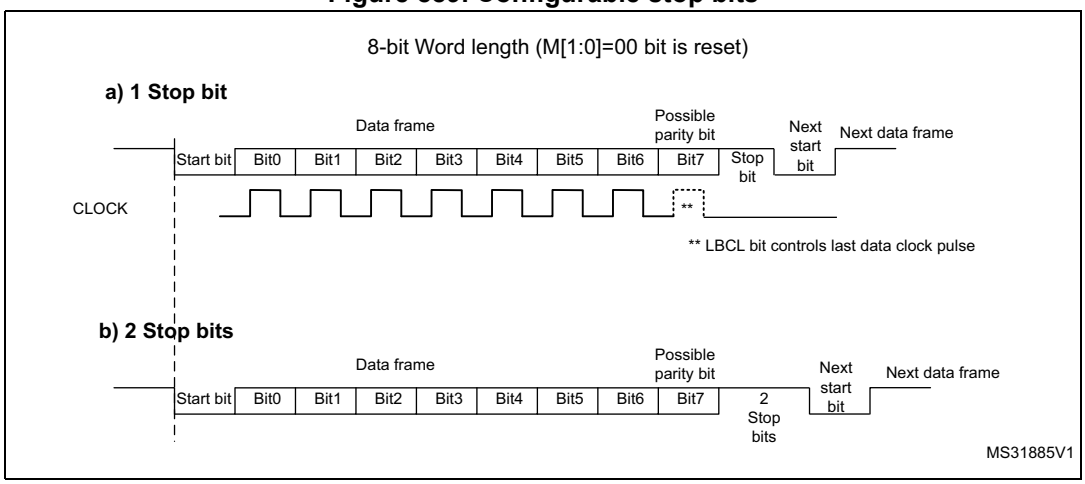
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire, and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 889. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 54.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when the TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  – the LPUART_TDR register is empty;
  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – The TXFIFO is not full;
  – The LPUART_TDR register is empty;
  – The next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

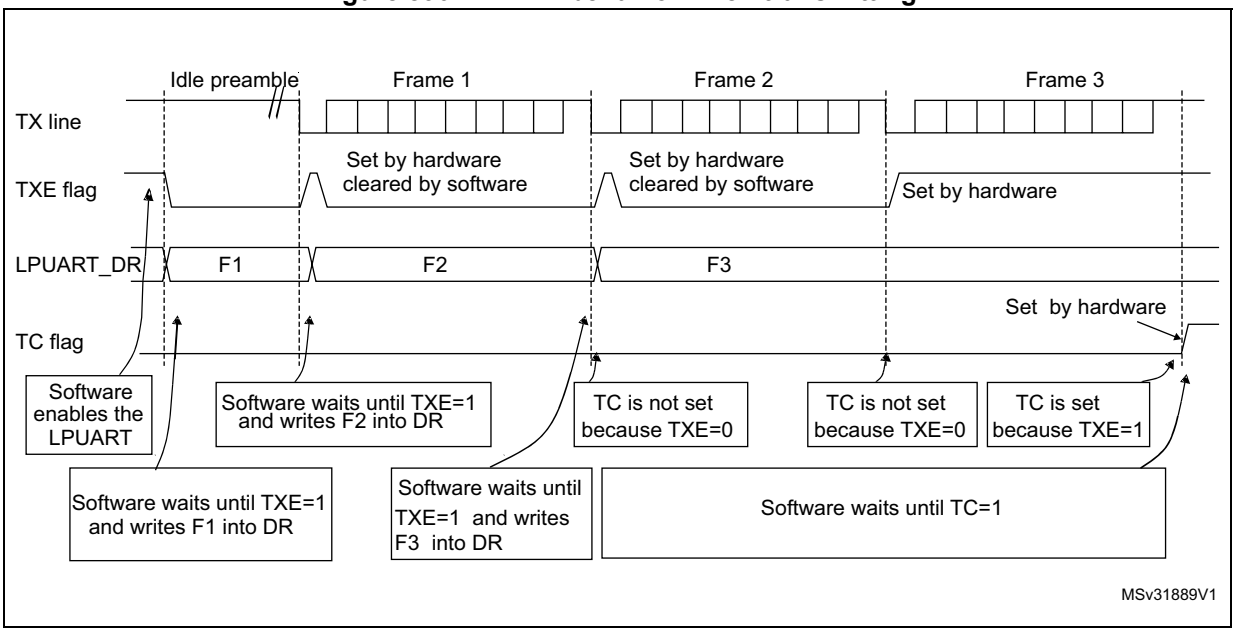
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 890: TC/TXE behavior when transmitting*).

Figure 890. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 888*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

54.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted with the least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 54.4.13: Continuous communication using DMA and LPUART*.
6. Set the RE bit LPUART_CR1. This enables the receiver, which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.

When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the receive data register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if the RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – The ORE bit is set;
  – The RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data cannot be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set;
  – The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *If RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the clock control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

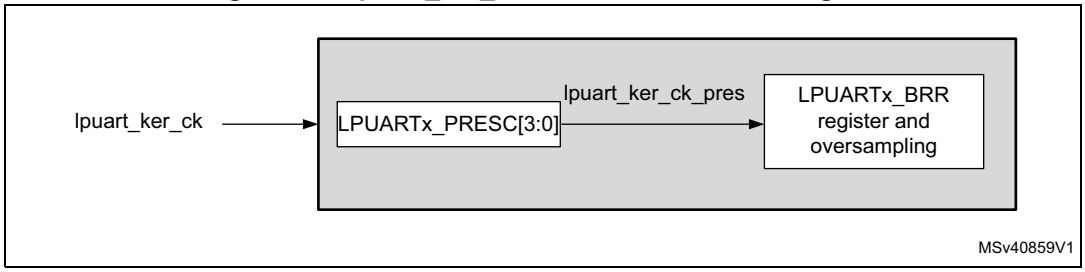
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 891. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a desynchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware.
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

54.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence, the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Table 628. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

Baud rate		lpuart_ker_ck_pres= 32.768 kHz		
S.No	**Desired**	**Actual**	**Value programmed in the baud rate register**	**% Error = (Calculated – Desired) B.rate / Desired B.rate**
1	0.3 kbaud	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

54.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 629*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 629. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = 10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00), 2 stop bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 629 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

54.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all nonaddressed receivers.

The nonaddressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in the LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

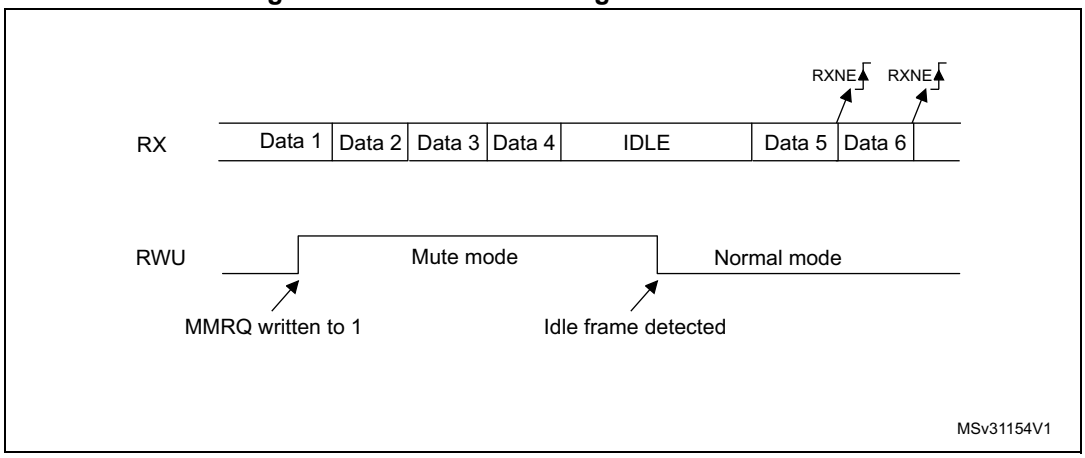
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 892*.

Figure 892. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7- or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address, which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

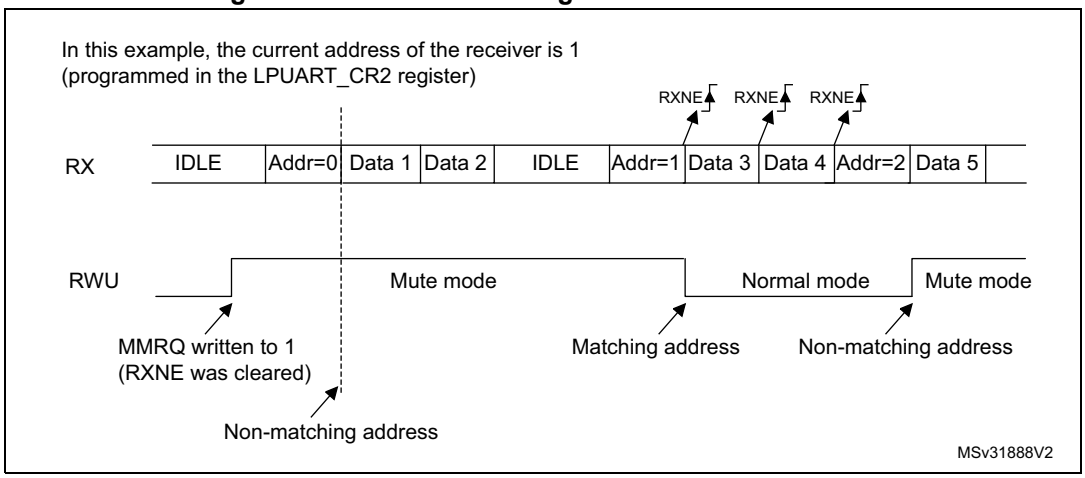
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 893*.

Figure 893. Mute mode using address mark detection



54.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 630*.

Table 630: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame, which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

54.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

54.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 54.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 54.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

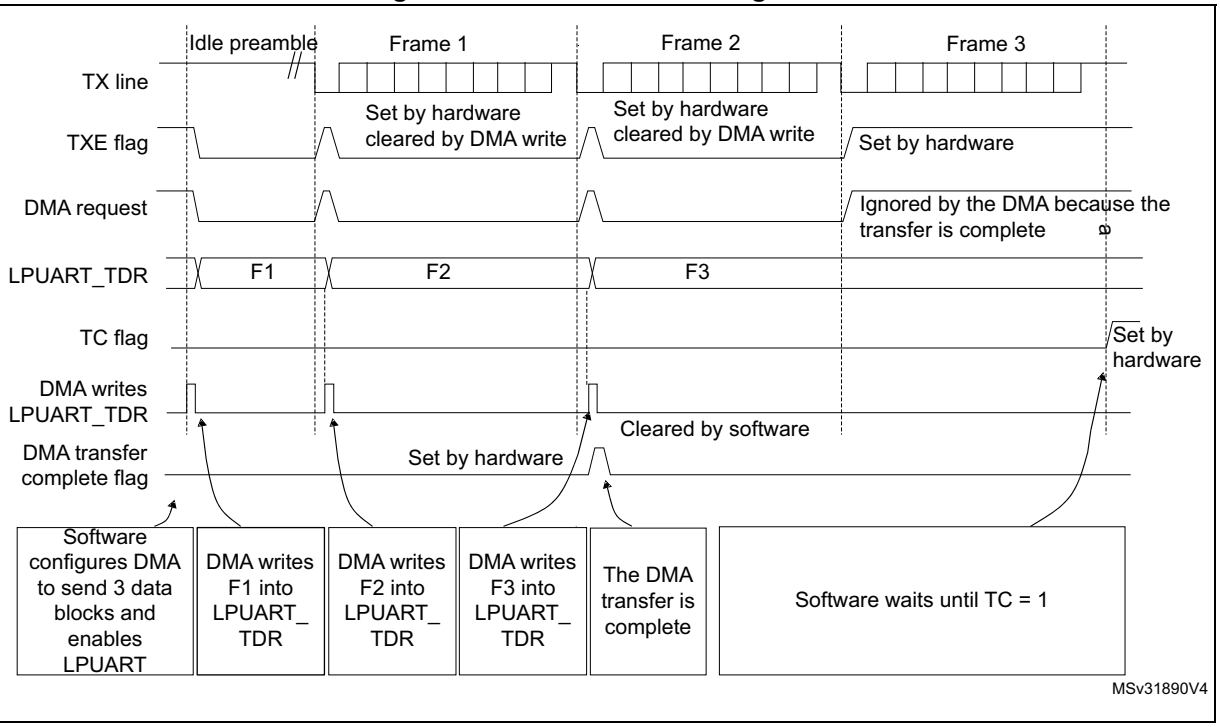
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 894. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

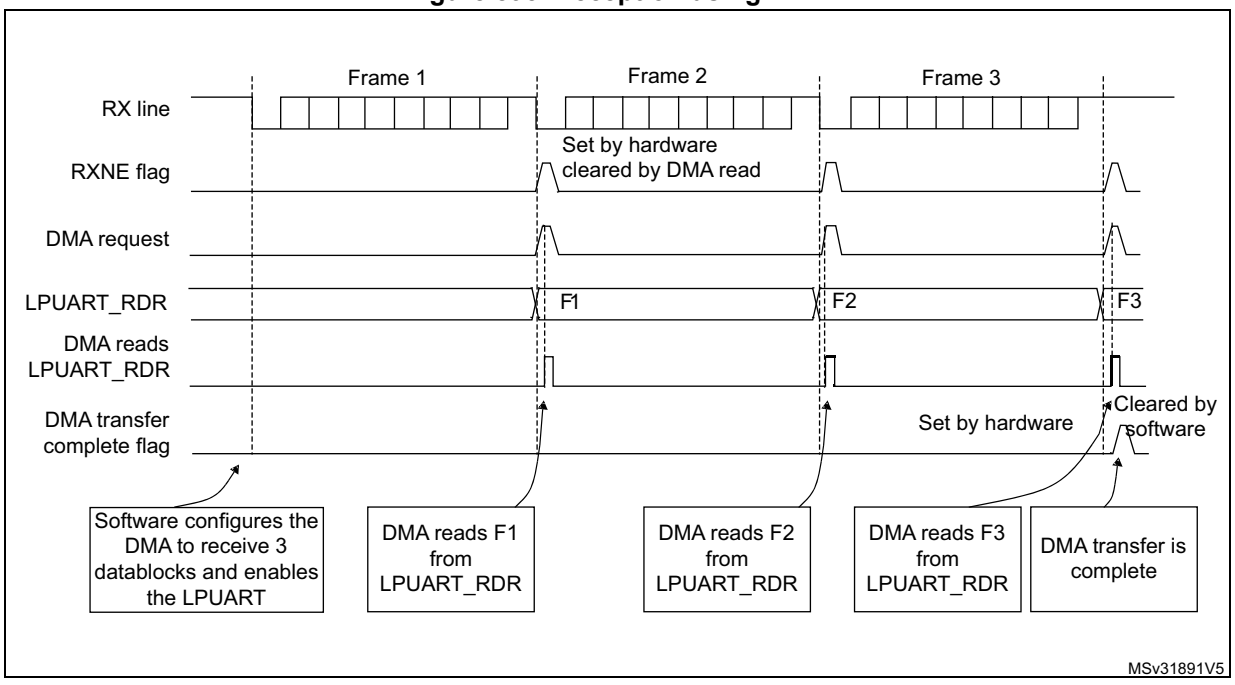
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 895. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

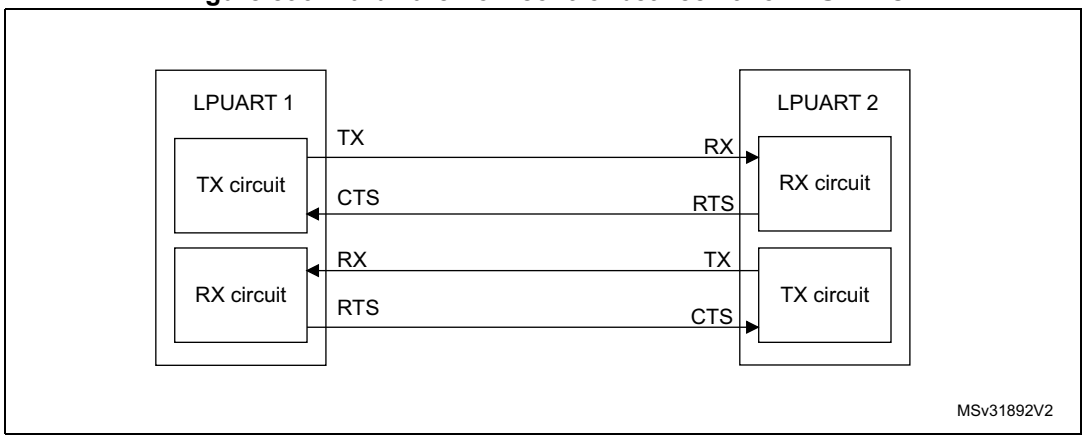
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

54.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. *Figure 896* shows how to connect two devices in this mode.

Figure 896. Hardware flow control between two LPUARTs

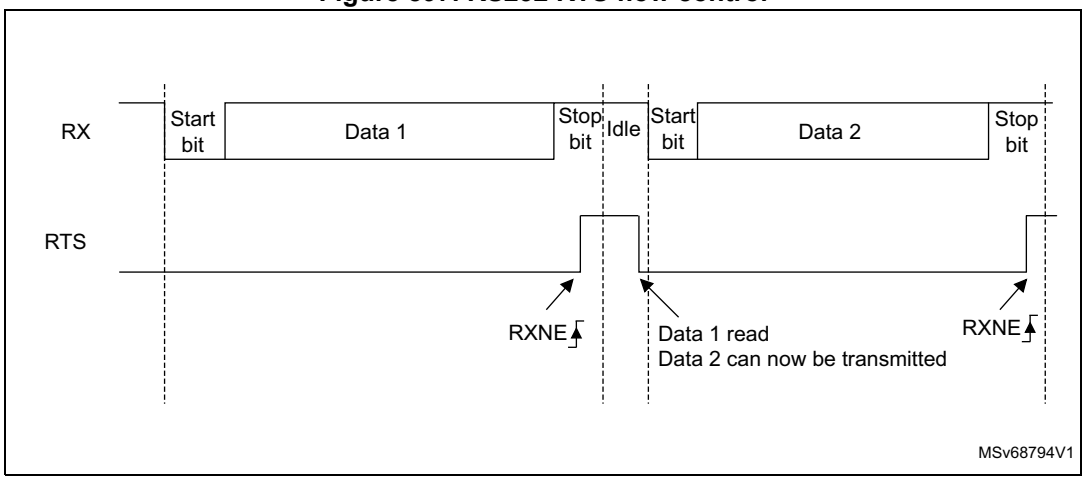


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 897* shows an example of communication with RTS flow control enabled.

Figure 897. RS232 RTS flow control



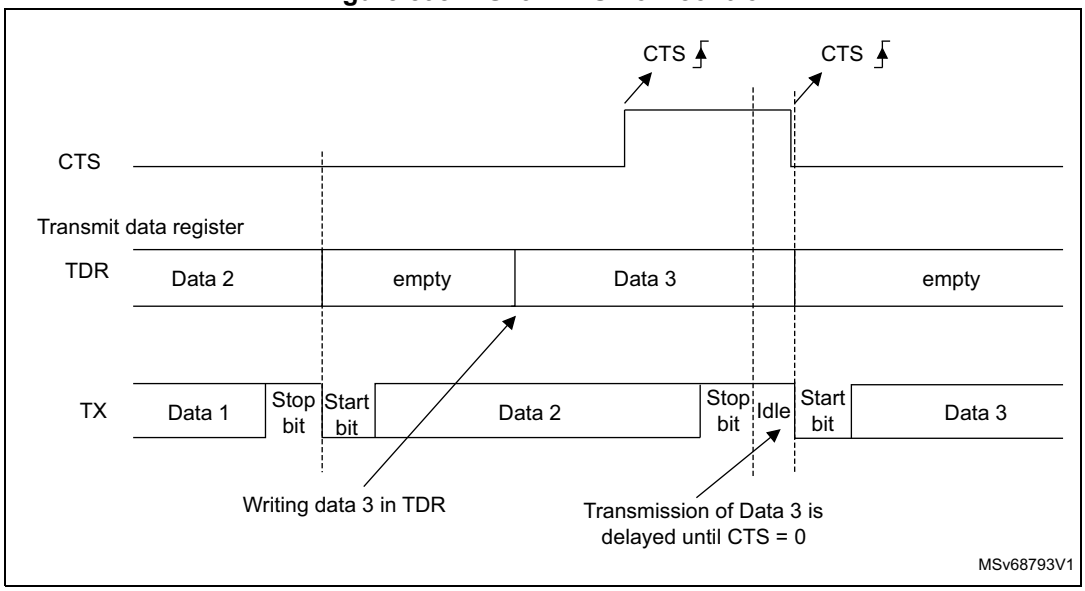
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 898* shows an example of communication with CTS flow control enabled.

Figure 898. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the deactivation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ($f_{lpuart_ker_ck_pres}$) cycles:

- The driver enable assertion time equals
  - (1 + (DEAT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEAT) x lpuart_ker_ck_pres, if P = 0
- The driver enable deassertion time equals
  - (1 + (DEDT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEDT) x lpuart_ker_ck_pres, if P = 0

where P = BRR[20:11]

54.4.15 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled.
The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.
When the lpuart_pclk is gated, the LPUART provides a wake-up interrupt (**lpuart_wkup**) if a specific action requiring the activation of the **lpuart_pclk** clock is needed:

- If FIFO mode is disabled

  luart_pclk clock has to be activated to empty the LPUART data register.

  In this case, the lpuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled

  luart_pclk clock has to be activated
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the lpuart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many

LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **lpuart_wkup** interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and the **lpuart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure that the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled upon exit from low-power mode.

When FIFO is enabled, the wake-up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the LPUART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode is used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the lpuart_ker_ck is switched OFF again, the MCU is not woken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 899 shows the behavior when the wake-up event is verified.

Figure 899. Wake-up event verified (wake-up event = address match, FIFO disabled)

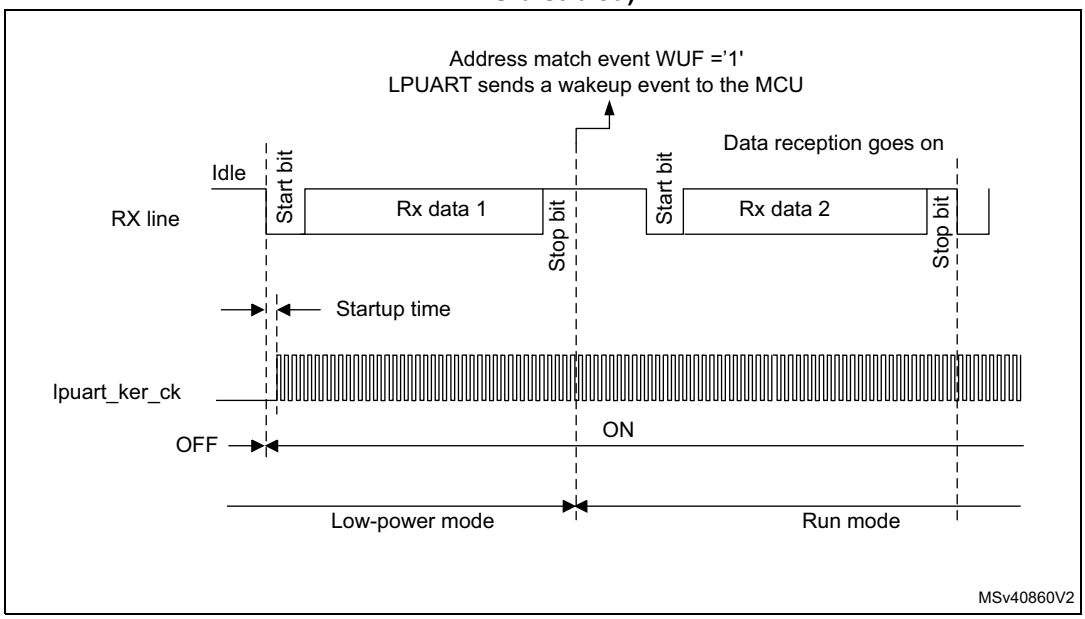
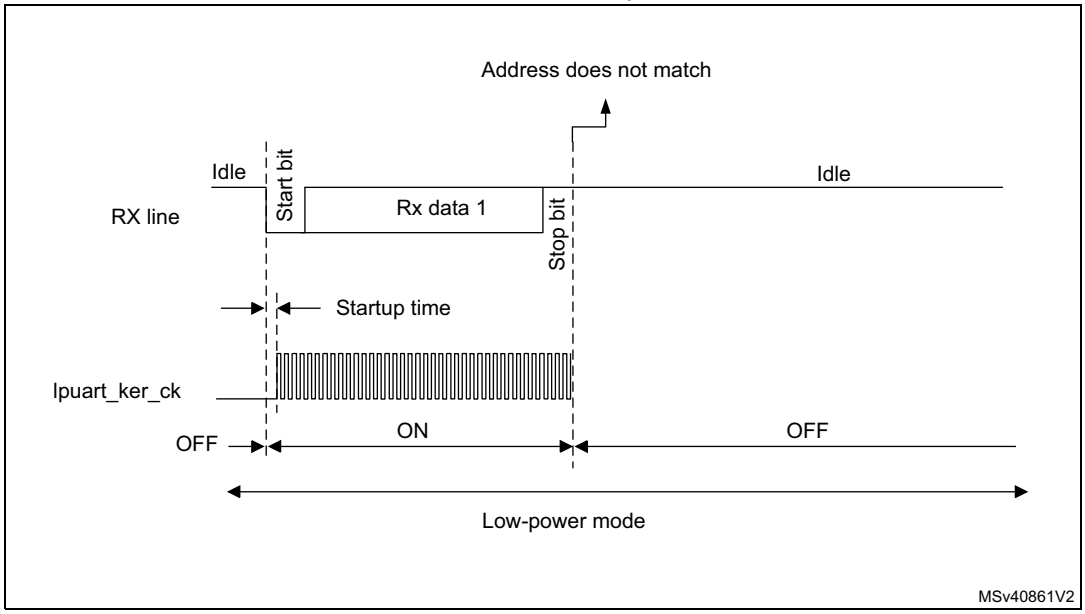


Figure 900 shows the behavior when the wake-up event is not verified.

Figure 900. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 54.4.9: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 629: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% – 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

54.5 LPUART in low-power modes

Table 631. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 54.3: LPUART implementation* to know if the wake-up from low-mode mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

54.6 LPUART interrupts

Refer to *Table 632* for a detailed description of all LPUART interrupt requests.

Table 632. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(3)		Write 1 in ORECF		No	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from low-mode mode feature. Refer to *Section 54.3: LPUART implementation* for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

54.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

54.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 53.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 54.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXFNF = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

54.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 53.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 54.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

54.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter is equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter is equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the LPUART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)
1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)
1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

54.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

54.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
									rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

54.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

54.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

54.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.
An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 54.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
0: Receiver in active mode
1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE = 1in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by a write to the LPUART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

54.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 54.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

54.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value.* Refer to *Section 54.3: LPUART implementation*.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

54.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 54.4.1: LPUART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

54.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 54.4.1: LPUART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

54.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation.

54.7.14 LPUART register map

Table 633. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	LPUART_CR1FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0	0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value										0	0	0					0	0	0	0		0	0	0	0	0			0				0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISRFIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISRFIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 633. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	LPUART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

55 Serial peripheral interface (SPI)

55.1 SPI introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex, and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

55.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection
- Multimaster or multislave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase with optional sampling delay when reading master data
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode

- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault, and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

55.3 SPI implementation

The table below describes the SPI implementation.

Table 634. SPI features

SPI feature	SPI1, SPI2 (full-featured instances)	SPI3[1] (full-featured instances)
Data and CRC size	Configurable from 4 to 32 bits	
CRC computation	CRC polynomial length configurable from 5 to 33 bits	
FIFO size	16x8 bits	
Number of data control (TSIZE)	Up to 65535	
I2S feature	Yes	No
Autonomous in Stop modes with wake-up capability	No	
Autonomous in LP-Stop and Standby modes with wake-up capability	No	

1. Available only on STM32G55x/56x devices

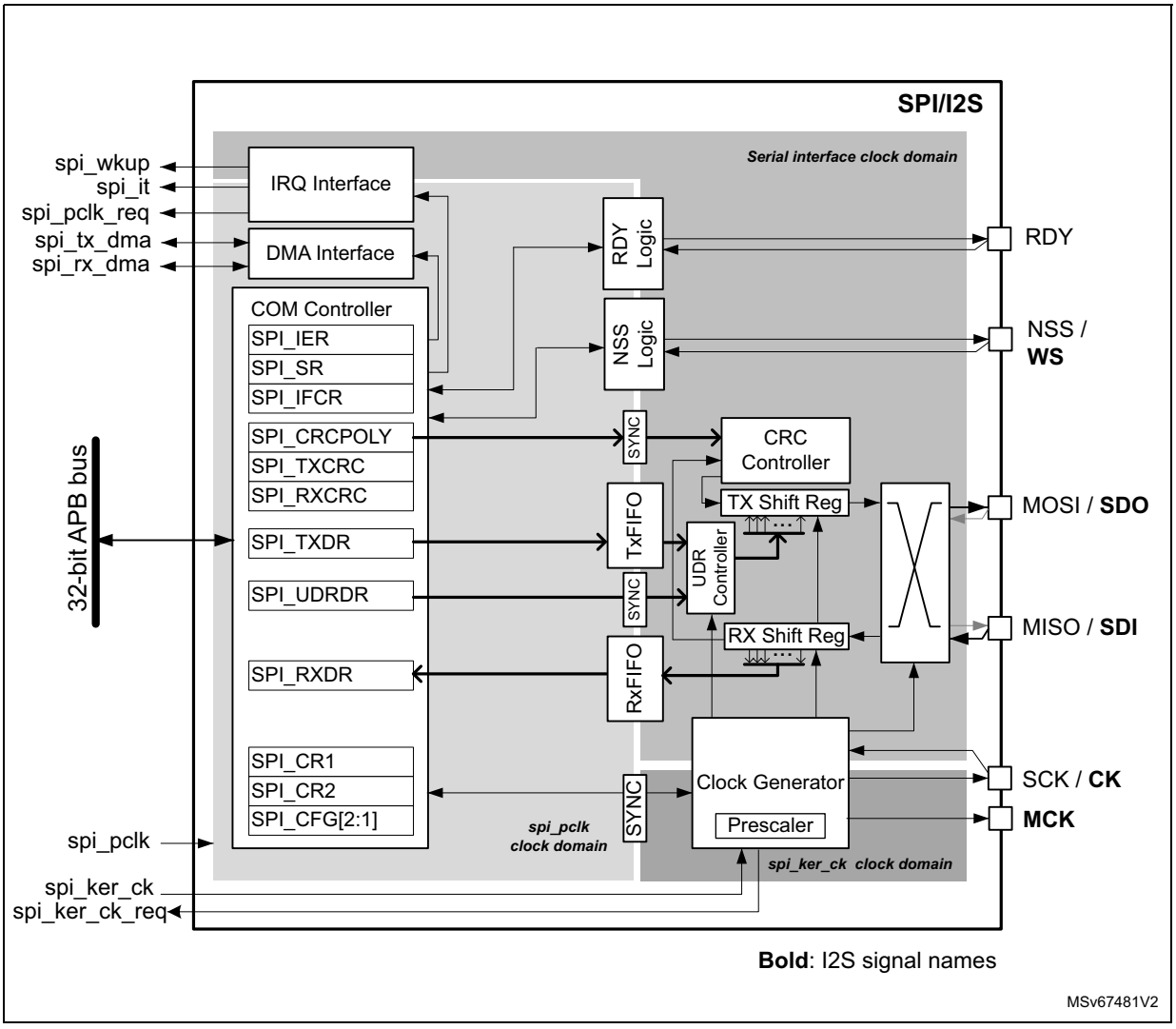
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 638: SPI wake-up and interrupt requests.

55.4 SPI functional description

55.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 901*.

Figure 901. SPI/I2S block diagram



The simplified scheme of *Figure 901* shows three fully independent clock domains:

- The **spi_pclk** clock domain
- The **spi_ker_ck** kernel clock domain
- The serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user

has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be set up correctly (for example when underrun or overrun is evaluated, see *Section 55.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 55.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

55.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 55.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the temporal suspension of the ongoing communication.

All these pins (except RDY) are shared in the I2S mode. This mode features an additional I2S specific MCK signal. For more details about I2S signals, see *Section 55.9.2*.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 635* and *Table 636* for the list of SPI input / output pins and internal signals.

Table 635. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
MISO/SDI(2)	Input/output	Master data input / slave data output
MOSI/SDO(2)	Input/output	Master data output / slave data input
SCK/CK	Input/output	Master clock output / slave clock input
NSS/WS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output
MCK	Output	I2S master frequency output

1. Refer to the section *Section 55.9.2: Pin sharing with SPI function* for details.
2. Functionality of MOSI/SDO and MISO/SDI pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 636. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wake-up interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request

55.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software slave select management) or three or four wires (with hardware slave select management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

55.4.4 Communications between one master and one slave

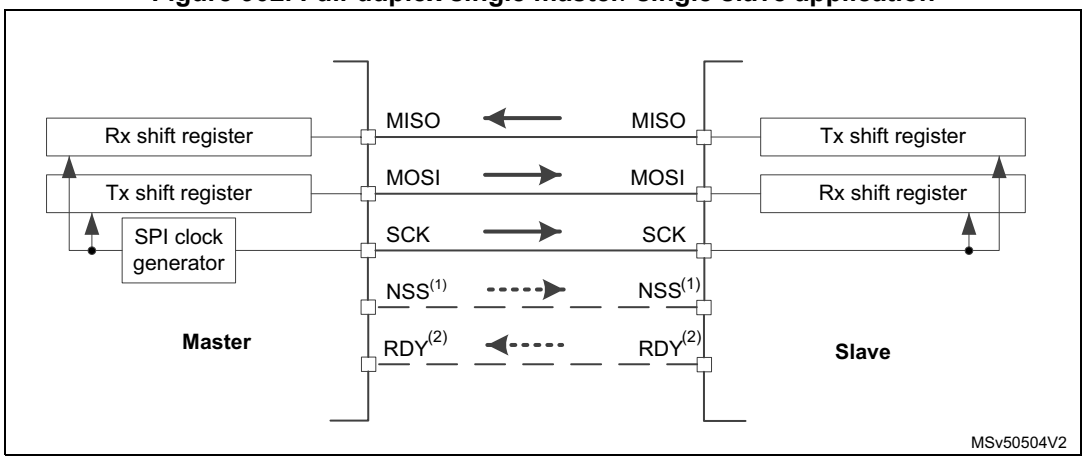
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to ensure the correct management of all the transferred data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 902. Full-duplex single master/ single slave application

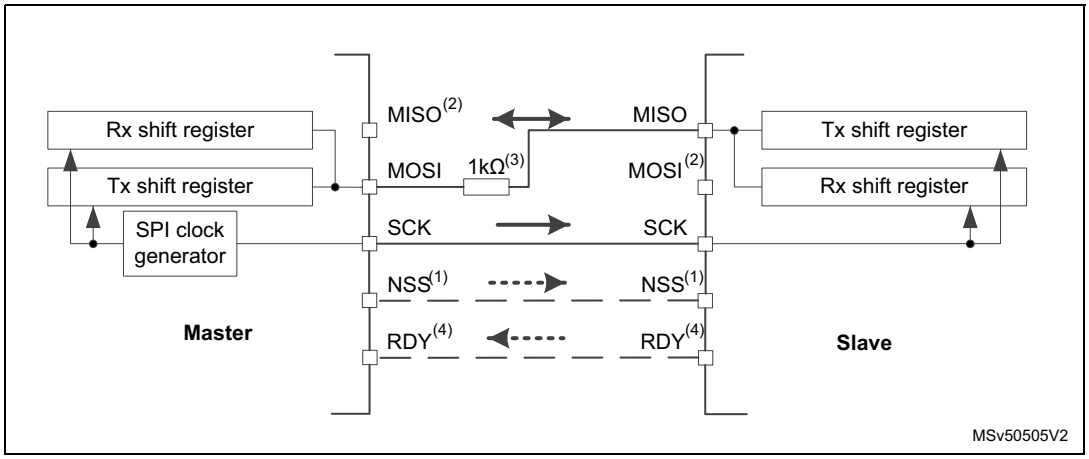


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 55.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross-connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing the direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 903. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 55.4.7*).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.
4. The RDY signal provided by the slave can be read by the master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pin pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the NSS pin (or software managed internal signal) is active (see *Section 55.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

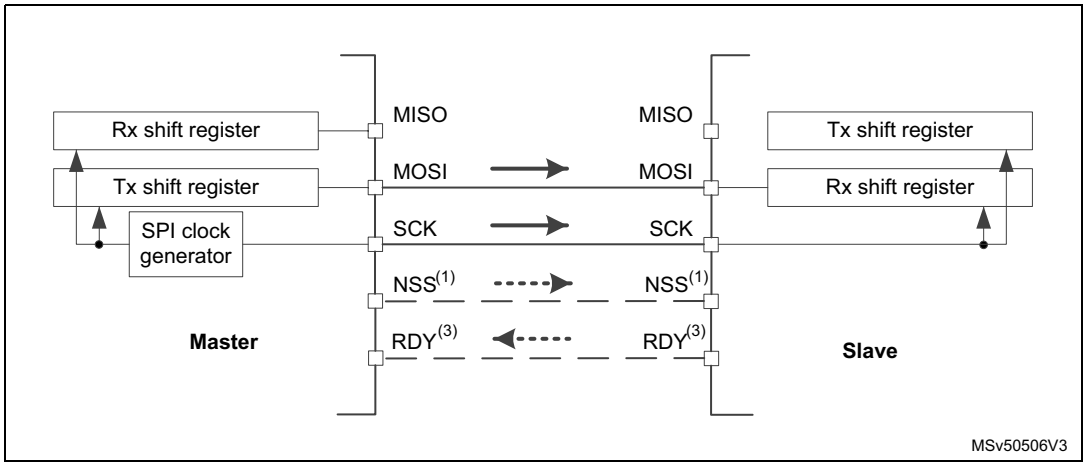
  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 55.4.7*).

Note: In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (this bit can only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transfer direction (bidirectional mode is enabled, while the

HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.

Figure 904. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 55.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

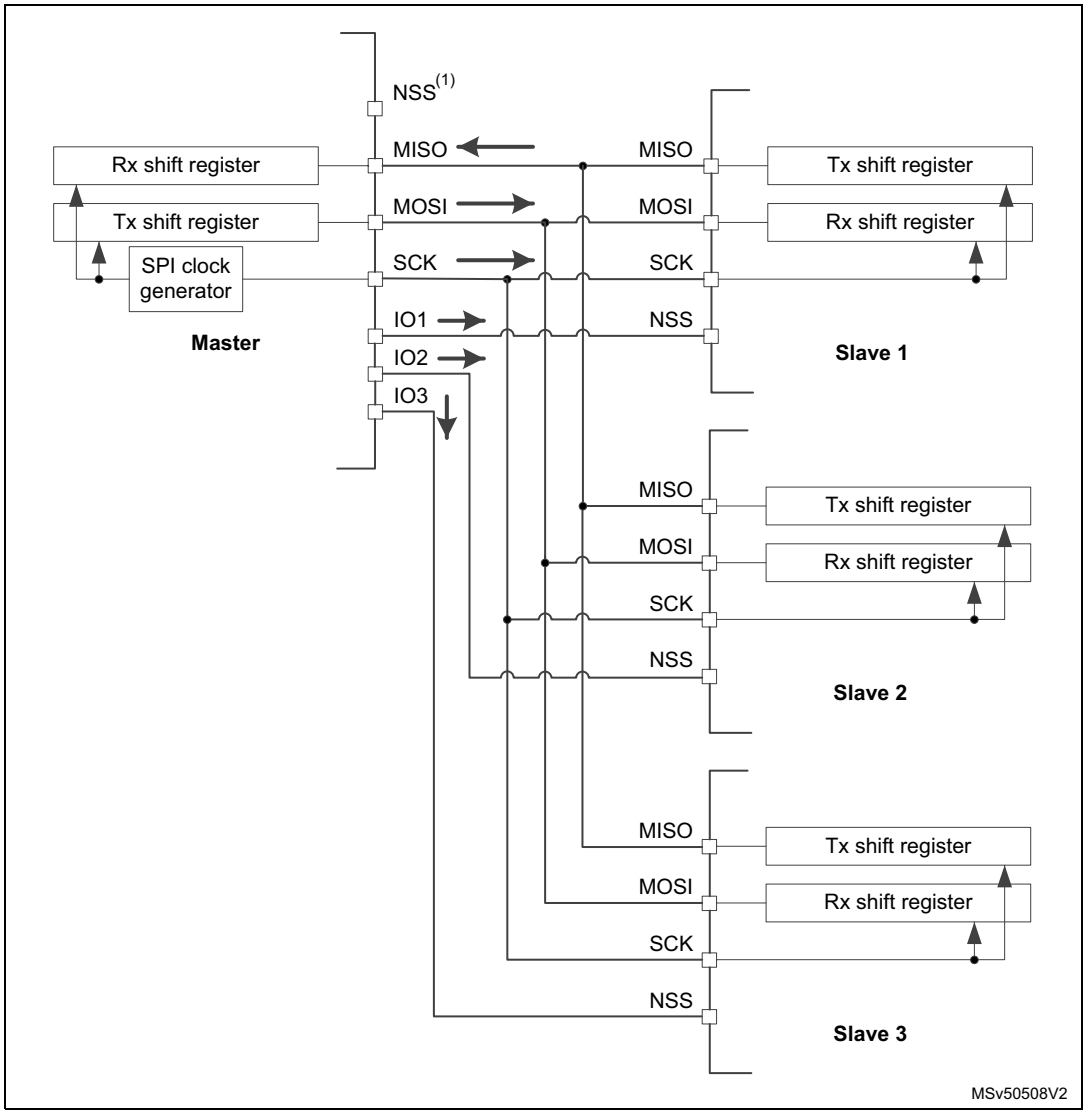
55.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 905.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within a single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 905. Master and three independent slaves connected in star topology



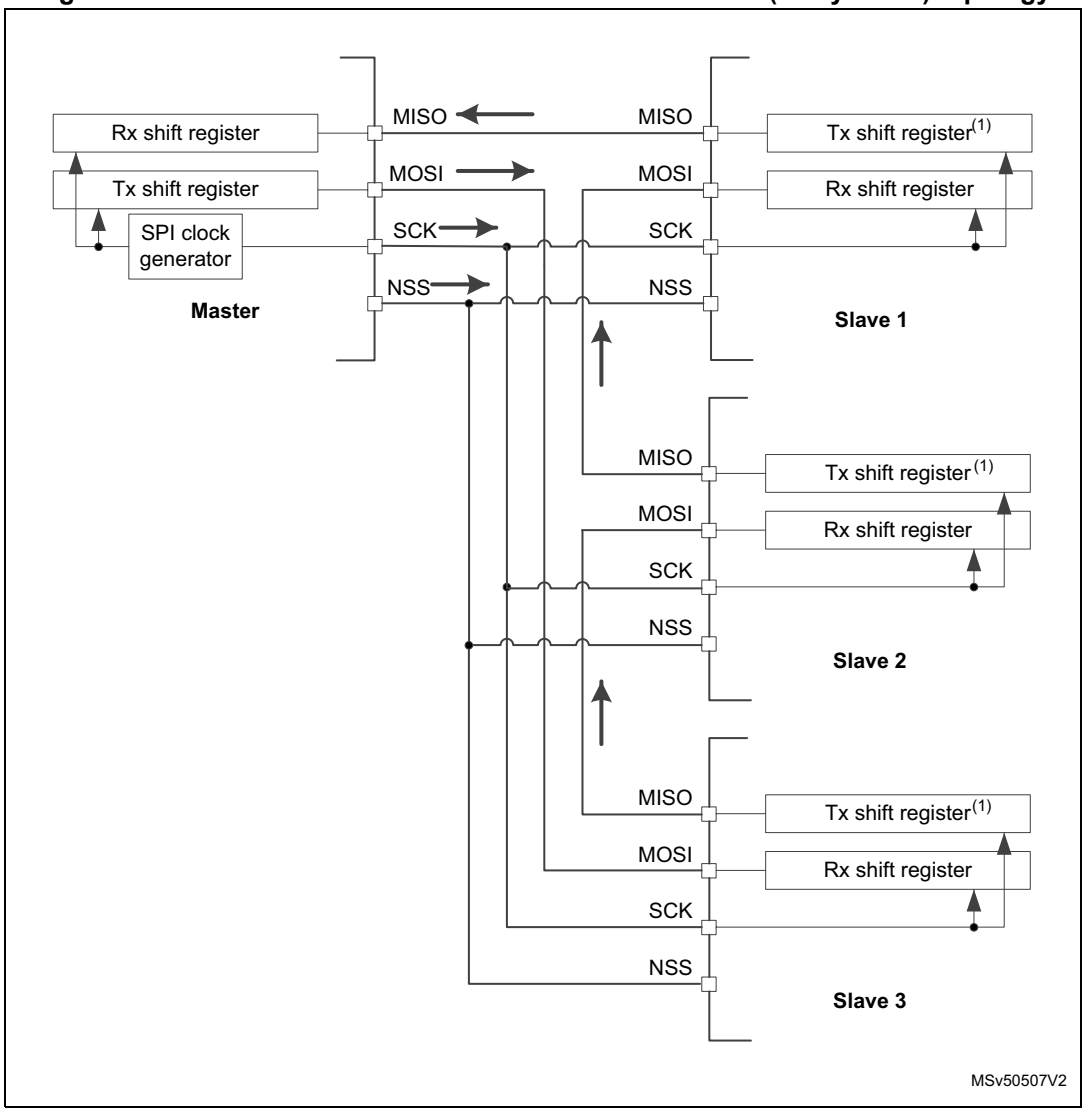
1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 55.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

The master can handle the SPI communication with all the slaves at the same time when a circular topology is applied (see *Figure 906*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (NSS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transferred by the master is equal to the number of slaves.

Then when a first data frame is transferred in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time.

Correspondingly, the last transferred data finishing the session is dedicated for the first slave node while its first outgoing data just reaches the master input, after its circling around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transferred between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating the last received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case, the underrun happens in fact after this first data frame is transferred. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 906. Master and three slaves connected in circular (daisy chain) topology



1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. The RDY signals can be read by the master optionally, either separately or configured as open drain outputs (while RDIOP = 0) and connected with a pull-up resistor as a common chain ready status overdriven by the slowest device.

55.4.6 Multimaster communication

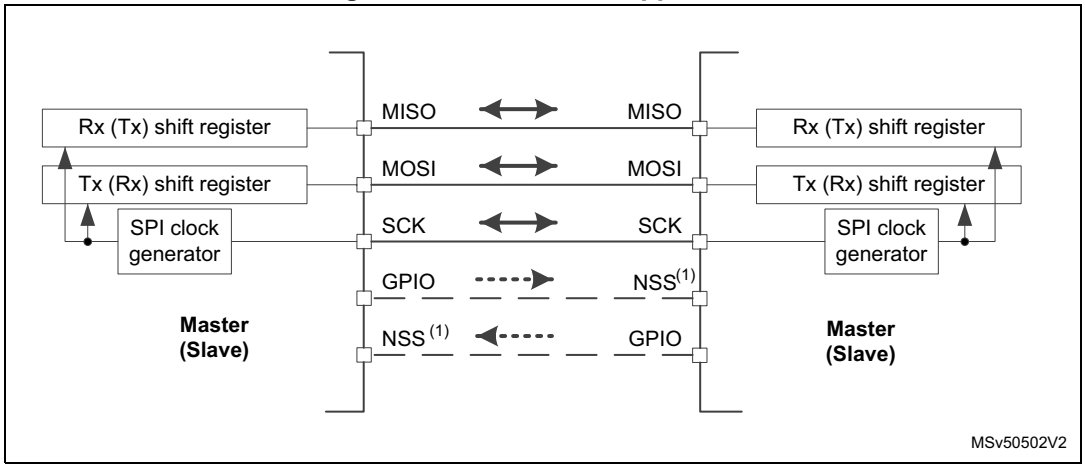
Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the NSS pin is configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is

complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 907. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control since the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

55.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault

evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

b) When SP = 001, a pulse is generated as defined by the TI mode.

c) When SSOM = 1, SP = 000 and MIDI > 1 the NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

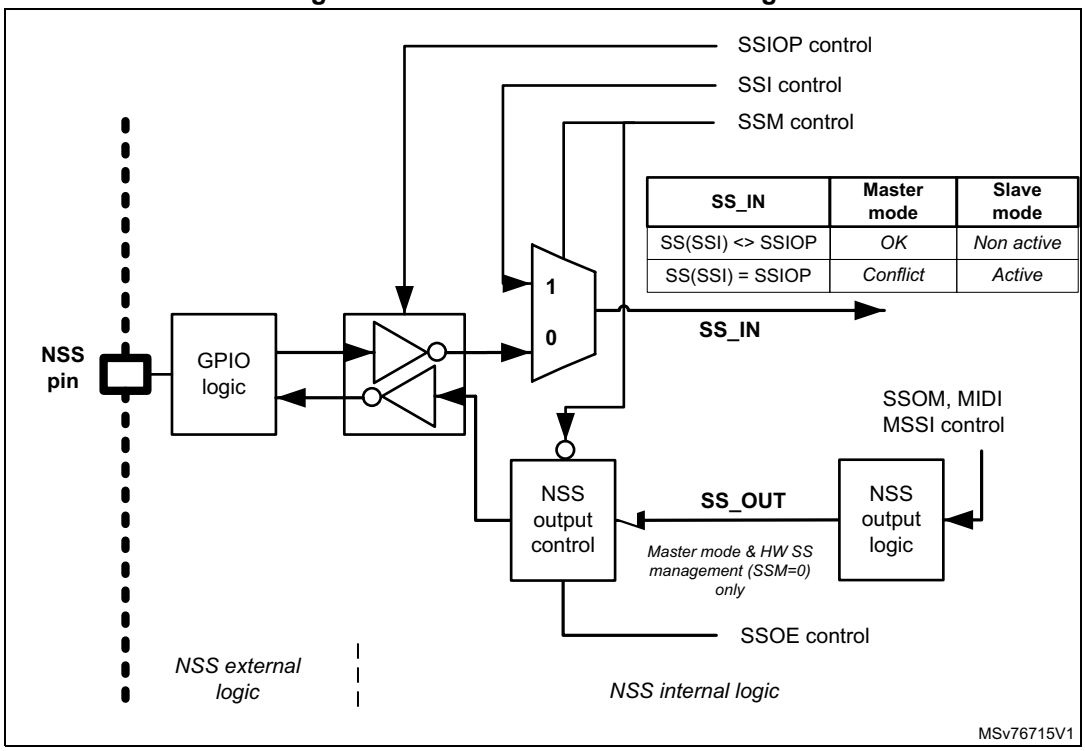
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

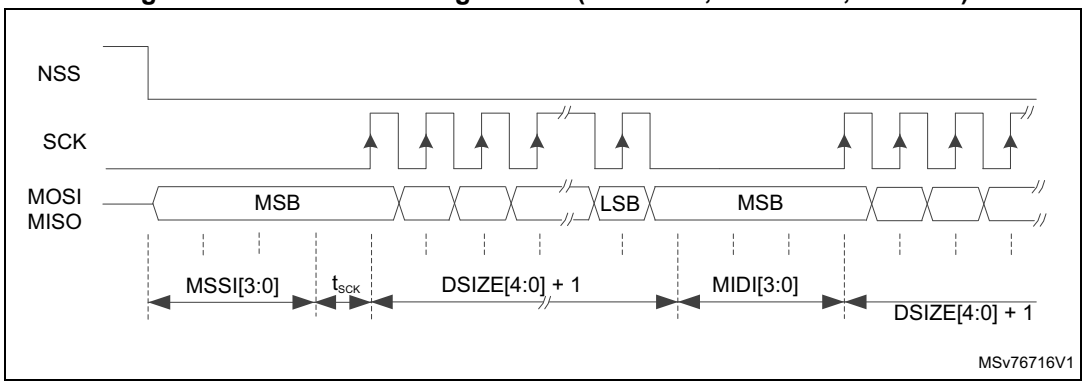
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 908. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 909*).

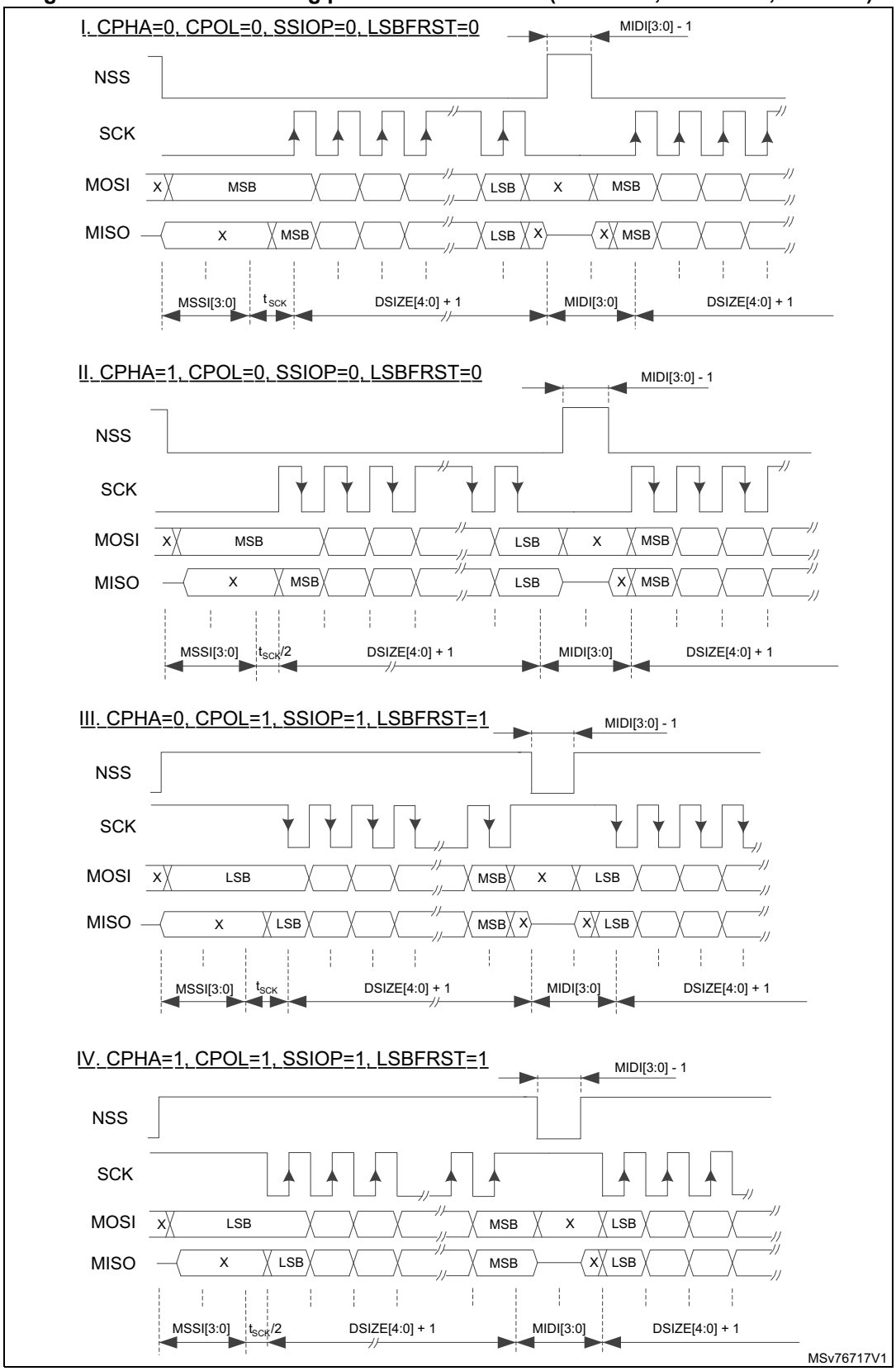
Figure 909. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 910*.

Figure 910. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

55.4.8 Ready pin (RDY) management

The status of the slave capability to handle data can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transfer. The reason can be that the slave TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transfer. This pessimistic logic is suppressed at the end of a transfer only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8 bits.

55.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity, and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships can be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

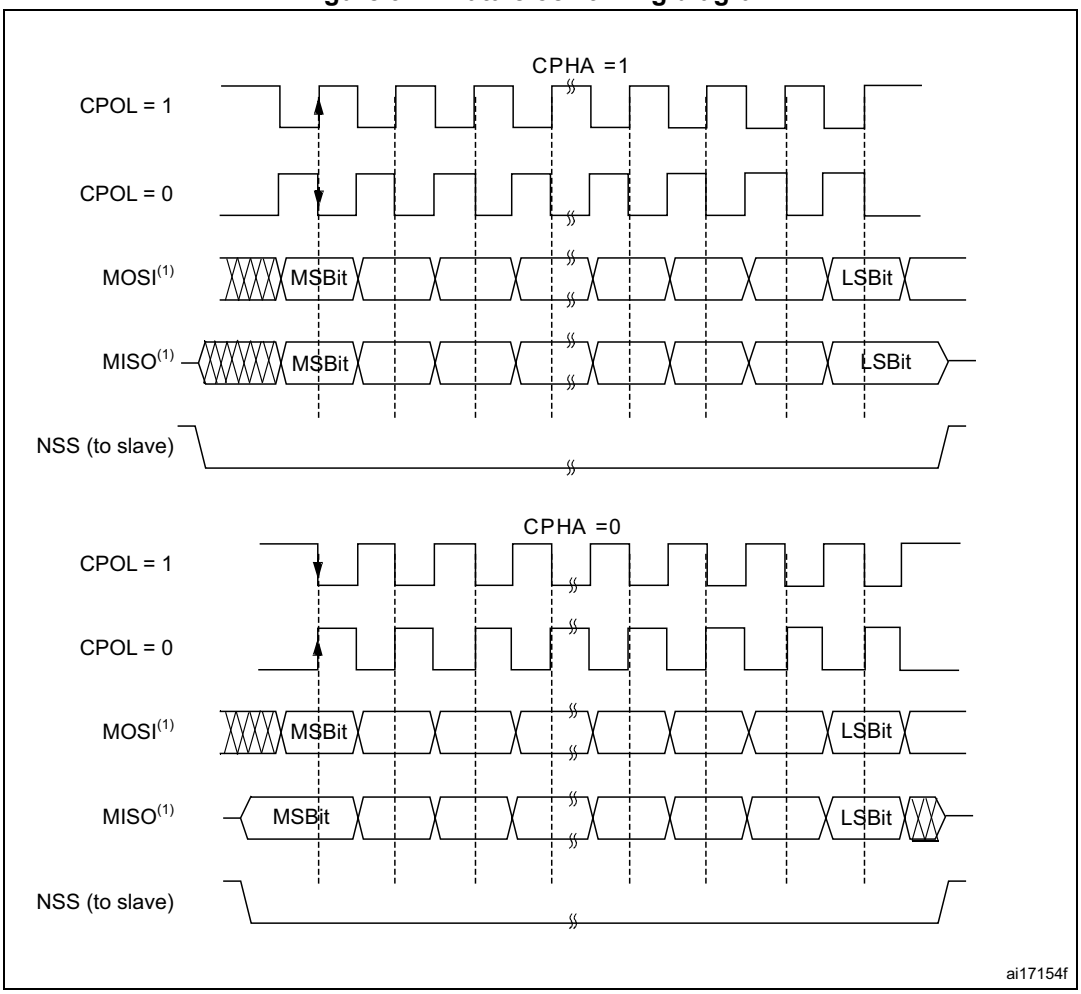
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 911*).

Figure 911 shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 911. Data clock timing diagram



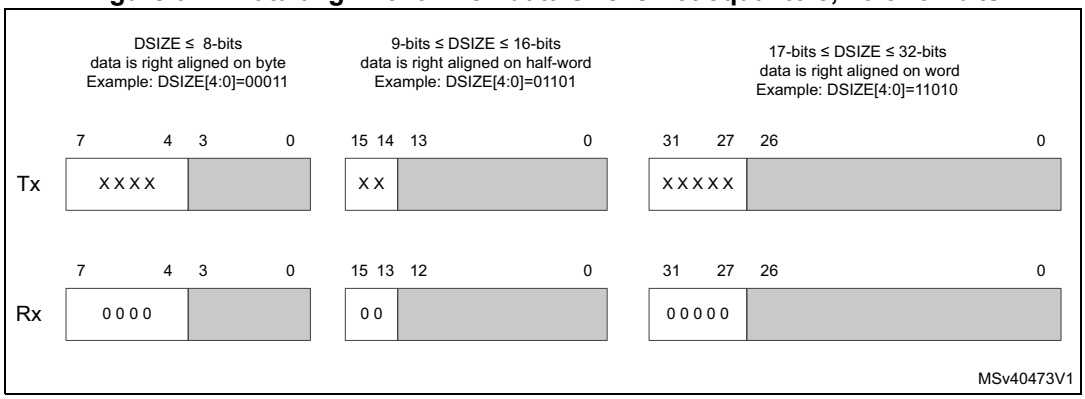
1. The order of data bits depends on the LSBFRST bit setting.
2. The input data sampling can be postponed by a half SCK cycle on the master MISO input (refer to the description of the DRDS bit of the SPI_CFG1 register).

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 55.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size (see *Figure 912*). These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 55.4.12: SPI data transmission and reception procedures*.

Figure 912. Data alignment when data size is not equal to 8, 16 or 32 bits



55.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured if CRCEN is set.
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI are not required in TI mode.

e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless of the peripheral enable. So, any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see *Section 55.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

55.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 55.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set, and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 55.4.14*.

55.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy

flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the

content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO

content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or

MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

55.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transfers may be corrupted.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE, and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer is still ongoing in the peripheral bus.

When TSIZE > 0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed.

TXC/EOT is set when the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the CSTART bit is cleared.

2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if the receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In slave mode, any ongoing data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transfer is complete.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done, for example by inserting a small software delay between EOT event occurrence and SPI disable action.

55.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set in the SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable the DMA Rx buffer in the RXDMAEN bit of the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable the DMA Tx buffer in the TXDMAEN bit of the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication, it is mandatory to follow these steps in order:

1. Disable DMA requests for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits of the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

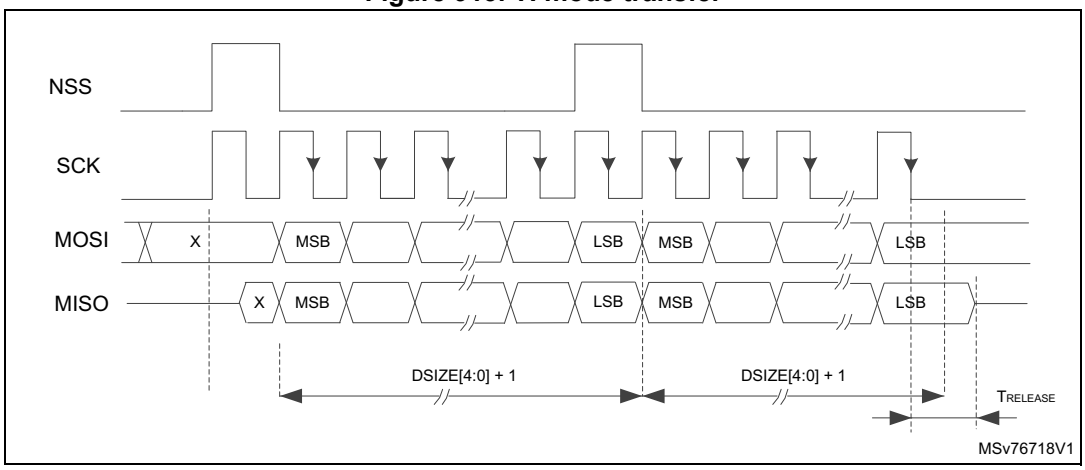
Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

55.5 SPI specific modes and control

55.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit order are fixed, so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI is not required when the SPI is in TI mode configuration. The NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 913*.

Figure 913. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi_ker_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi_ker_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

55.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the corresponding interrupt enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all

data transmitted subsequently are lost. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

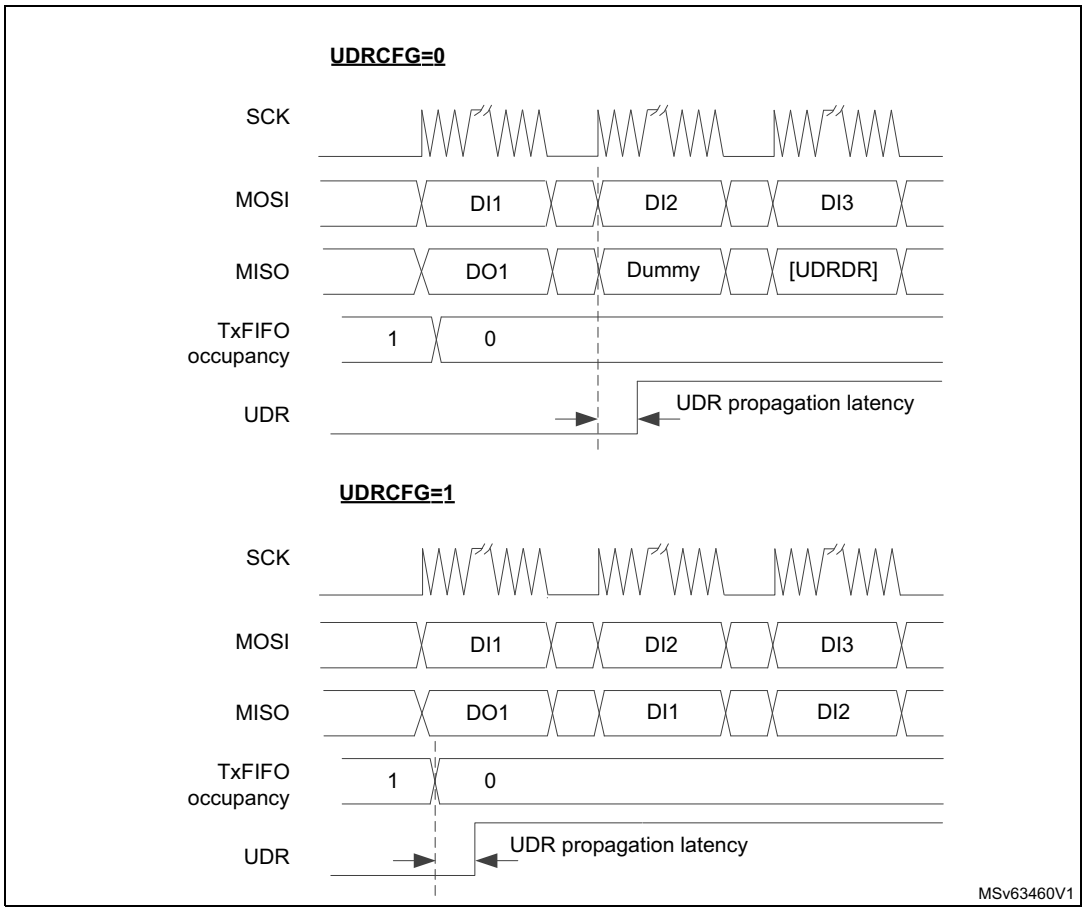
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 914*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 906*). Assuming that TxFIFO is not empty when the master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the last received data immediately.

The standard transmission is reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 914. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero until the MODF bit is set. This disables the SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device, the MODF bit cannot be set except as the result of a previous multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

55.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 55.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size. Its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in the case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits of the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is complete, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag of the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI of the SPI_CR1 register.

There is no need to reset these registers between communications, and they are not cleared when SPI is disabled. Their contents are not reliable while communication is ongoing; they become reliable only after the CRCE or EOT event is asserted, in particular.

55.6 SPI in low-power modes

Table 637. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the SPI registers is kept.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 55.3: SPI implementation* for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

55.7 SPI interrupts

Table 638 gives an overview of the SPI events capable of generating interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to the description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 638. SPI wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability[1][2]
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC[3]		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 55.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

55.8 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length can be 16, 24 or 32 bits[a]
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, overrun, and frame errors
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - F_{MCK} = 256 x F_{WS} for all I2S modes
  - F_{MCK} = 128 x F_{WS} for all PCM modes

Note: F_{MCK} is the master clock frequency and F_{WS} is the audio sampling frequency.

55.9 I2S functional description

55.9.1 I2S general description

The block diagram shown on *Figure 901* also applies for I2S mode.

The SPI/I2S block can work in I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA, and parts of interrupt signaling are shared with the SPI function. The low-power mode function is also available in I2S mode (refer to *Section 55.6: SPI in low-power modes* and *Section 55.10: I2S interrupts*).

a. Not always available, refer to *Section 55.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

55.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: serial data output (mapped on the MOSI pin) to transmit the audio samples in master, and receive the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 2970*.
- SDI: serial data input (mapped on the MISO pin) to receive the audio samples in master, and transmit the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 2970*.
- WS: word select (mapped on the NSS pin). This is the frame synchronization. It is configured as output in master mode, and as input in slave mode.
- CK: serial clock (mapped on the SCK pin). This is the serial bit clock. It is configured as output in master mode, and as input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: master clock (mapped separately) is used when the I2S is configured in master mode.

55.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = '1'), some bitfields are no longer relevant, and must be forced to a specific value to guarantee the behavior of the I2S/PCM function. *Table 639* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 639. Bitfields usable in PCM/I2S mode

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S control register 1 (SPI_CR1)	IOLOCK, CSUSP, CSTART, SPE	Other fields set to their reset values
SPI/I2S control register 2 (SPI_CR2)	-	Set to reset value
SPI/I2S configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI/I2S configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI_SR)	SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S receive data register (SPI_RXDR)	The complete register	-

Table 639. Bitfields usable in PCM/I2S mode (continued)

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI/I2S transmitter CRC register (SPI_TXCRC)	-	
SPI/I2S receiver CRC register (SPI_RXCRC)	-	
SPI/I2S underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

55.9.4 Slave and master modes

The SPI/I2S block supports master and slave modes for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

To improve the robustness of the SPI/I2S block in slave mode, the peripheral resynchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB-justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected

Note: This resynchronization mechanism is not available for the I2S LSB-justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

55.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits of the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frame formats are supported:

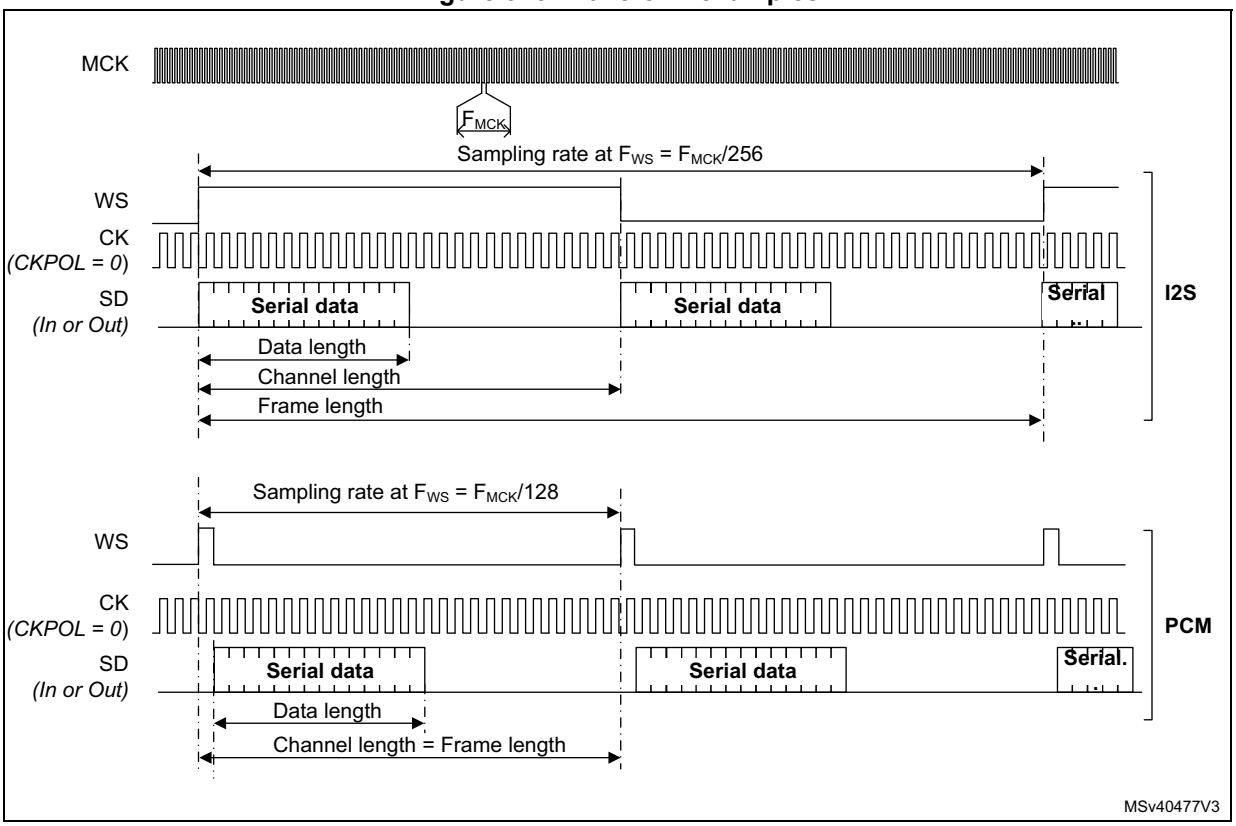
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel(a)
- 32-bit data packed in a 32-bit channel(a)

In PCM master mode, three frame formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 915. Waveform examples



For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to *Section : Serial data line swapping* for details about the direction control of serial data lines.

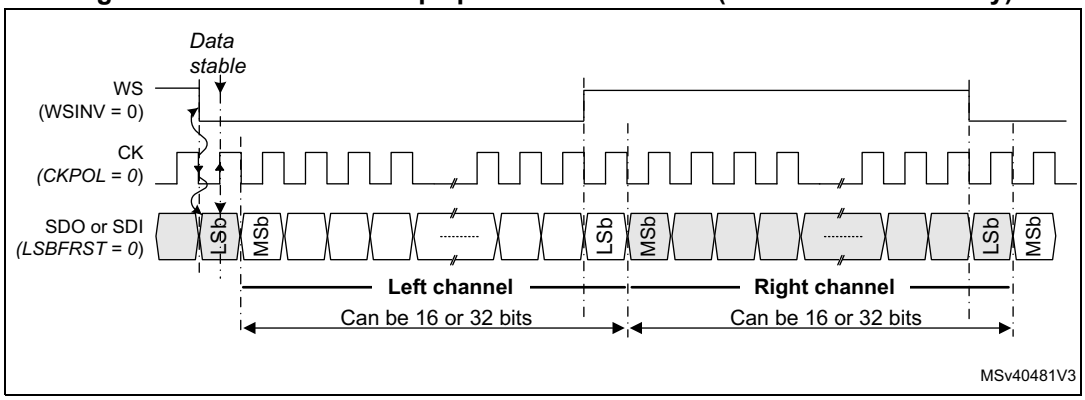
I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

a. Not always available, refer to *Section 55.3: SPI implementation* to check if 24 and 32-bit data widths are supported.

Figure 916. Master I2S Philips protocol waveforms (16/32-bit full accuracy)



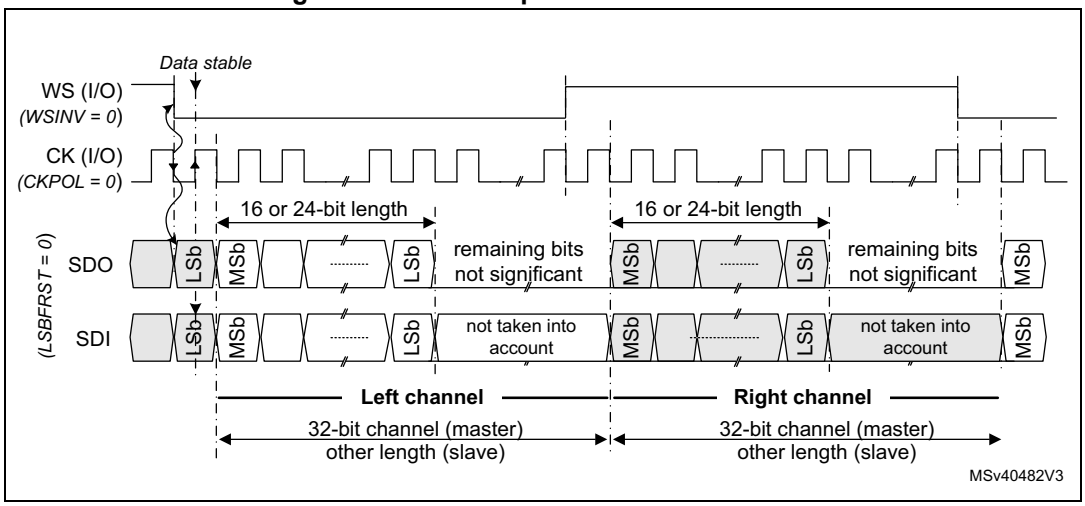
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* for the supported data sizes.

CKPOL is cleared to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

Figure 916 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

Figure 917. I2S Philips standard waveforms



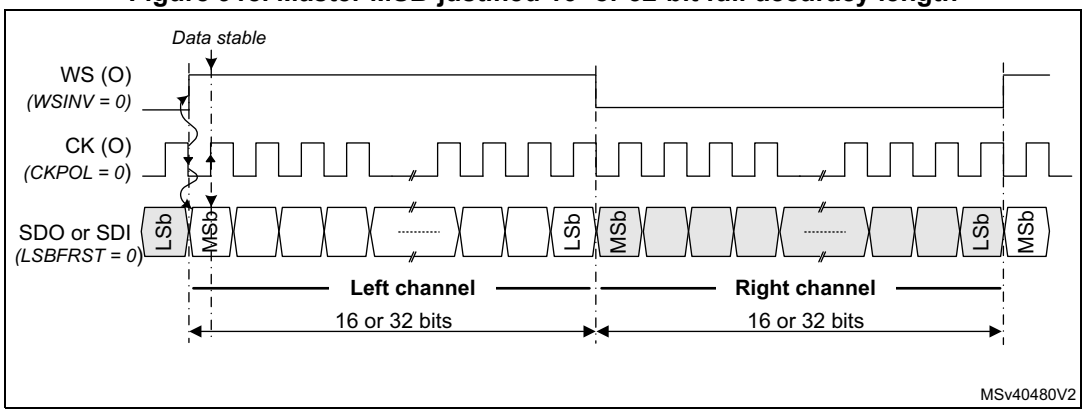
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB-justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high and the right channel if WS is low.

Figure 918. Master MSB-justified 16- or 32-bit full-accuracy length

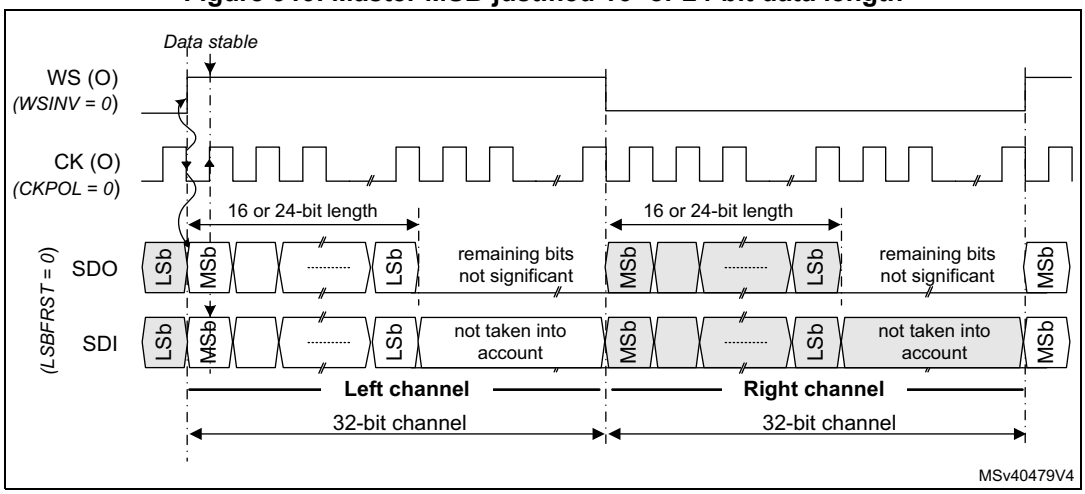


1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* to check the supported data size.

CKPOL is cleared to match the I2S MSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

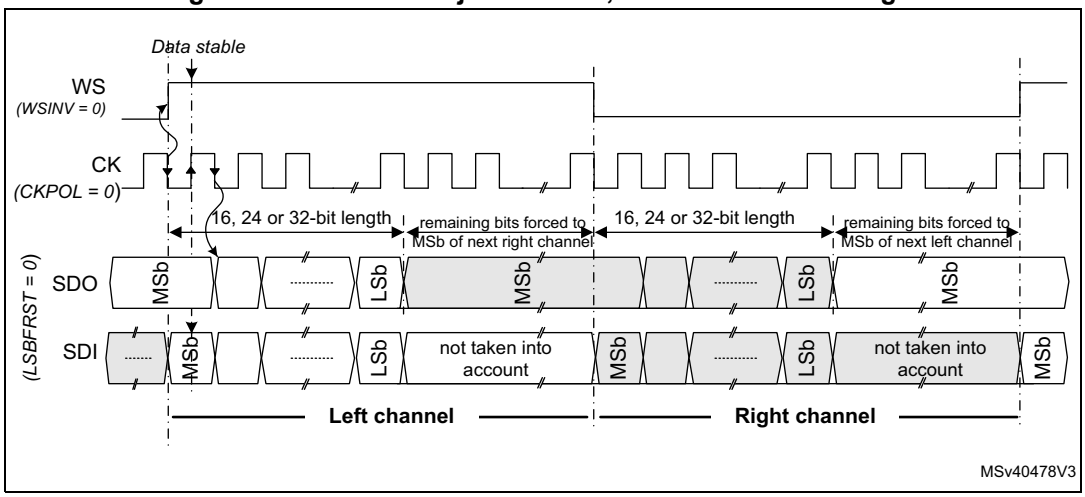
Figure 919. Master MSB-justified 16- or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit mode, the remaining bits are forced to the value of the first bit of the next data to be generated to avoid timing issues (see *Figure 920*).

Figure 920. Slave MSB-justified 16-, 24- or 32-bit data length



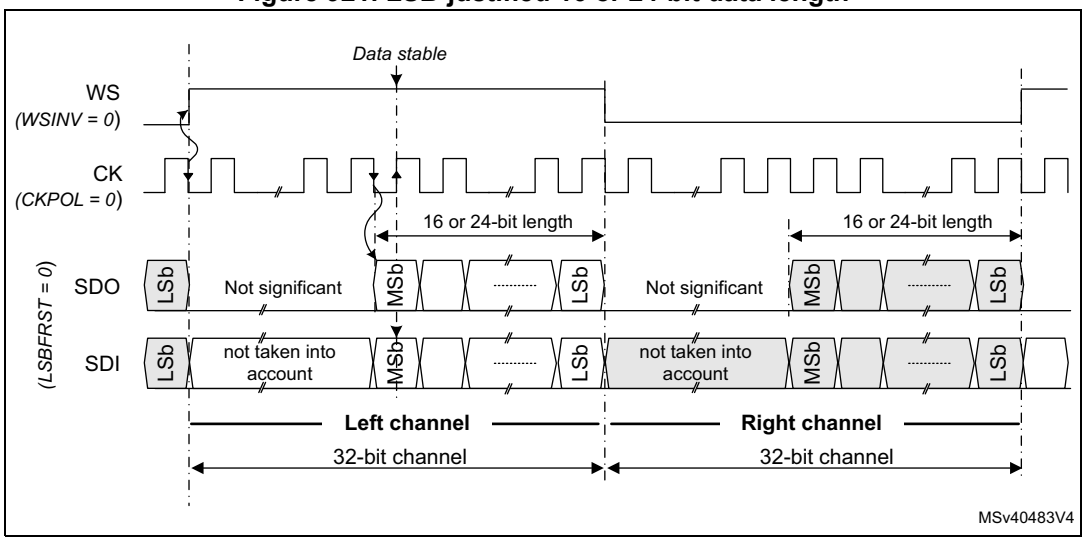
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* to check the supported data size.

LSB-justified standard

This standard is similar to the MSB-justified standard in master mode (no difference for the 16- and 32-bit full-accuracy frame formats). The LSB-justified 16- or 32-bit full-accuracy format gives similar waveforms to MSB-justified mode (see *Figure 918*) because the channel and data have the same length.

Note: In the LSB-justified format, only 16- and 32-bit channel lengths are supported in master and slave mode. This is because it is not possible to transfer properly the data if the channel length is not known by the transmitter and receiver side.

Figure 921. LSB-justified 16 or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* to check the supported data size.

CKPOL is cleared in order to match the I2S LSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit of SPI_I2SCFGR register.

In PCM long frame:

- The assertion time of the WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

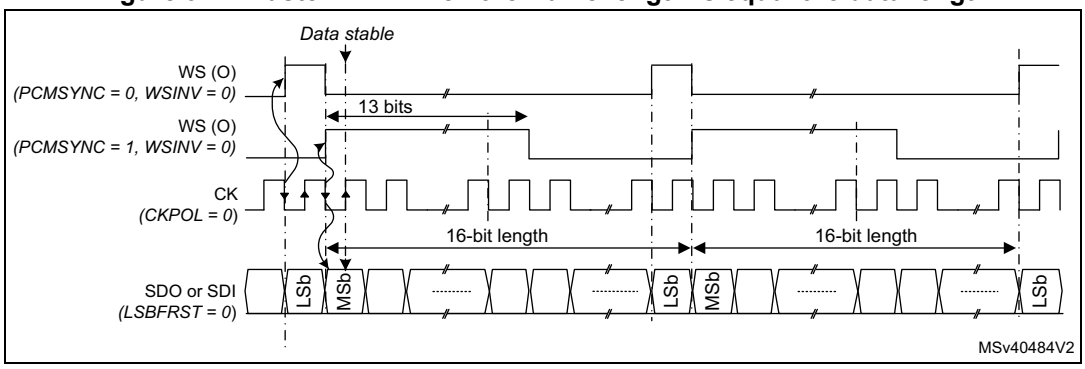
In PCM short frame:

- The assertion time of the WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:

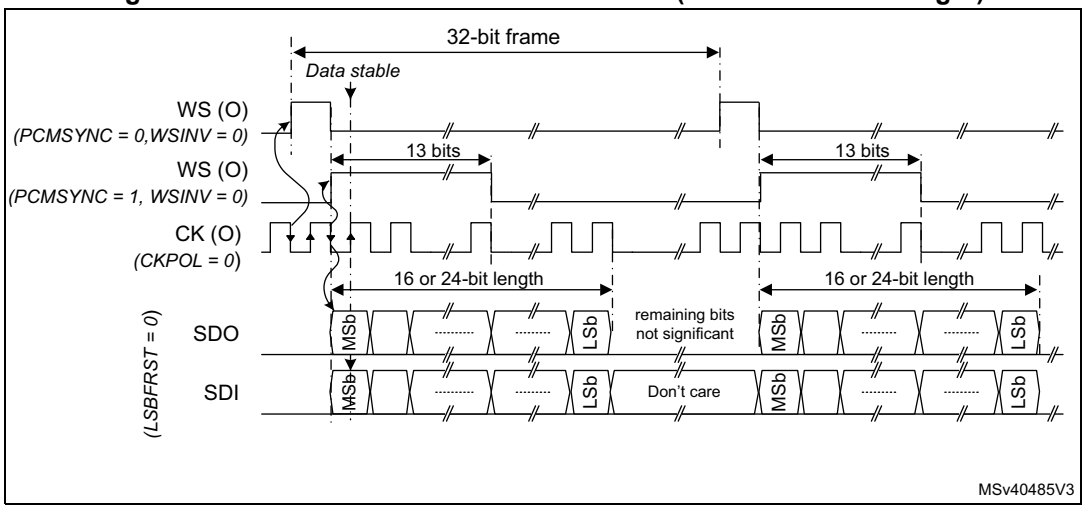
- The first data bit is MSb or LSb depending on the LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to the WSINV bit. See *Control of the WS inversion* for information concerning the handling of WS signal.

Figure 922. Master PCM when the frame length is equal the data length



A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

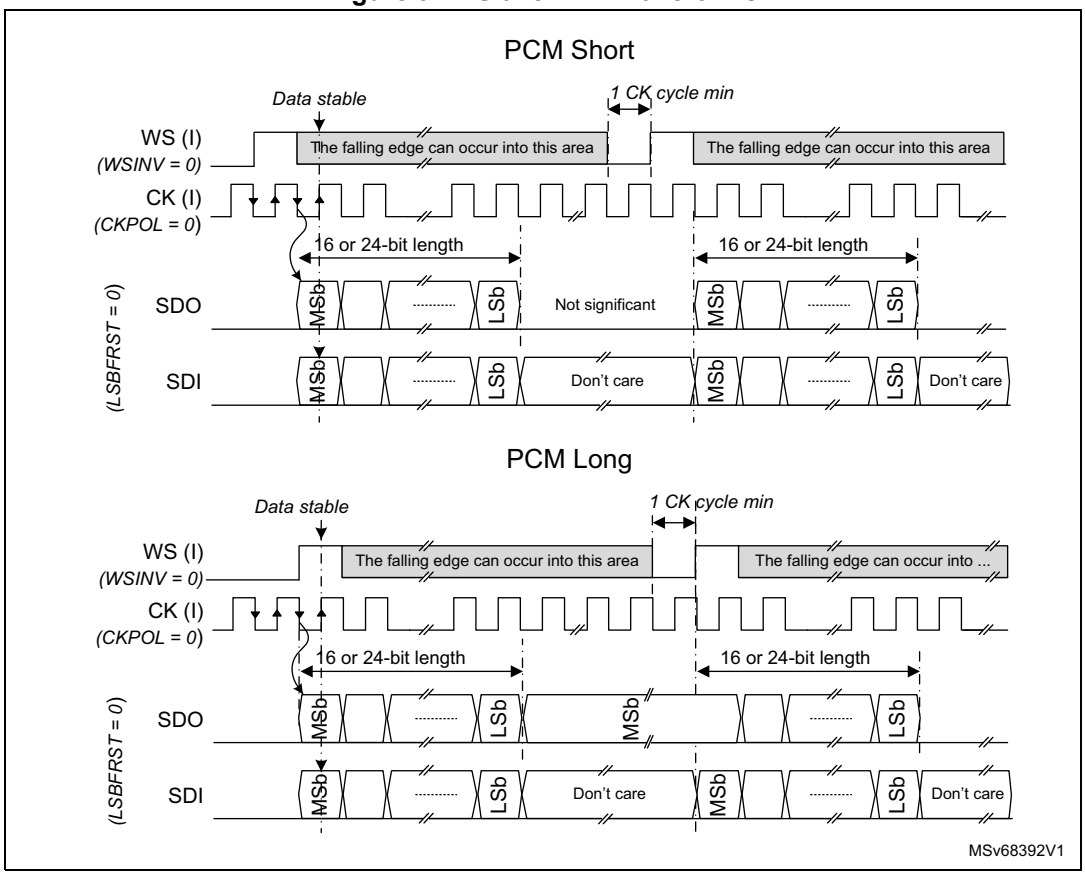
Figure 923. Master PCM standard waveforms (16 or 24-bit data length)



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* to check the supported data size.

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 924*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 924. Slave PCM waveforms



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* to check the supported data size.

Note: In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits is forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

55.9.6 Additional serial interface flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for the I2S LSB-justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI/I2S configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB-justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB-justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located in the *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI/I2S configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 640 shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (that is inactive level). Note that the level of WS also depends on the protocol selected.

Table 640. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: The bit AFCNTR must not be set when the SPI is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of *SPI/I2S configuration register 2 (SPI_CFG2)*, and on the slave/master mode. *Table 641* gives details on this feature.

Table 641. Serial data line swapping

Direction	IOSWP	Master mode		Slave mode	
		Input line	Output line	Input line	Output line
RX	0	SDI	-	SDO	-
	1	SDO	-	SDI	-
TX	0	-	SDO	-	SDI
	1	-	SDI	-	SDO
Full-duplex	0	SDI	SDO	SDO	SDI
	1	SDO	SDI	SDI	SDO

55.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write from/to the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

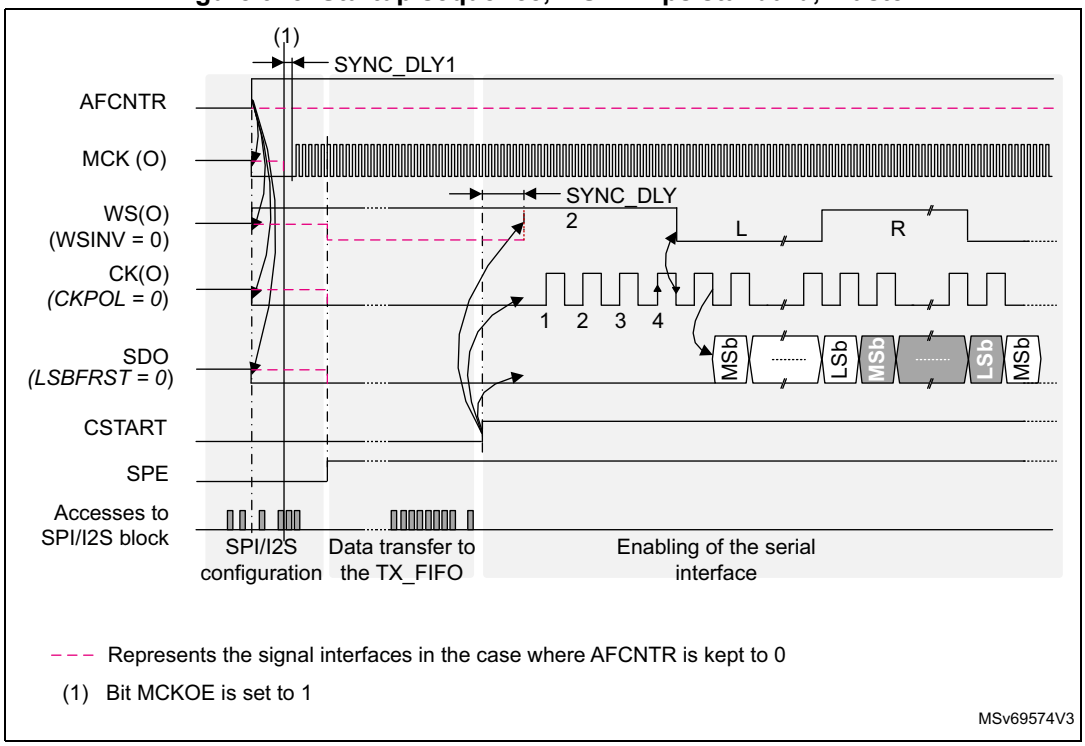
When the application wants to use the SPI/I2S block the user has to proceed as follows:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 55.9.16* for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC, and UDRC bits of the SPI_IFCR register. Note that if the flag SUSP is not cleared (via the SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO.
5. Set bit CSTART to activate the serial interface.

As shown in *Figure 925*, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1 and the TxFIFO is not empty. Note that the CK bit clock is activated four rising edges before

the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 925. Startup sequence, I2S Philips standard, master



2. When MCKOE is set, the MCK is generated after a synchronization delay (SYNC_DLY1). For more information, see MCK generation in *Section 55.9.9: Clock generator*.
3. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set.

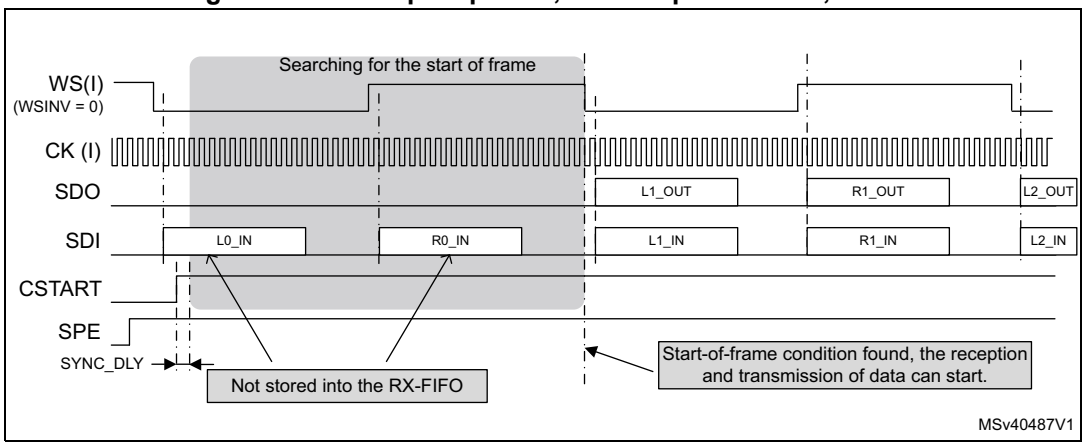
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about three periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For the I2S Philips standard, the start-of-frame condition is a falling edge of the WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of the WS signal. The transmission/reception starts at the rising edge of WS for MSB-aligned protocol. The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 926 shows an example of startup sequence in I2S Philips standard, slave mode.

Figure 926. Startup sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after two periods of CK clock (SYNC_DLY).

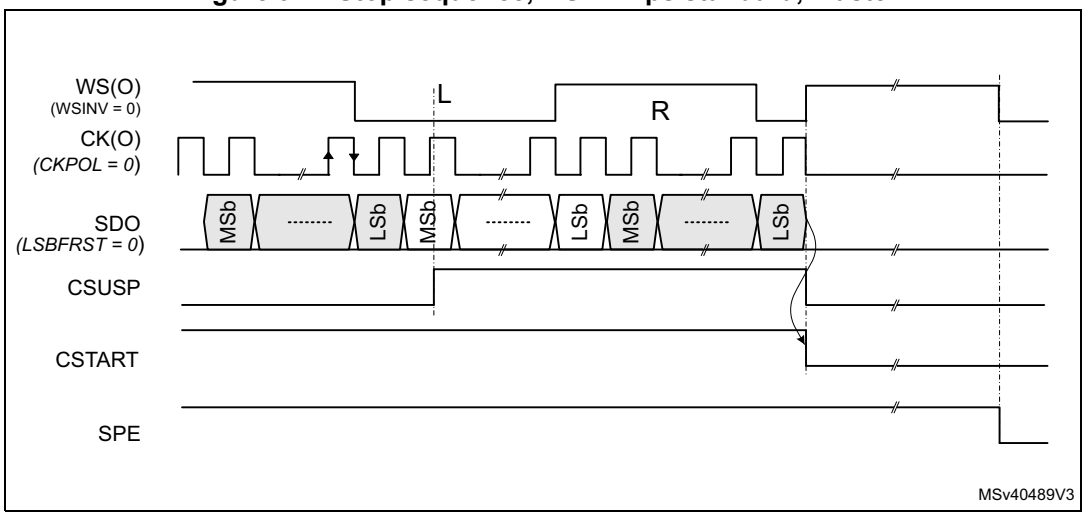
55.9.8 Stop sequence

The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

Figure 927 shows an example of stop sequence in the case of master mode. The CSUSP bit is set during the transmission of the left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear the SPE bit.

Figure 927. Stop sequence, I2S Philips standard, master

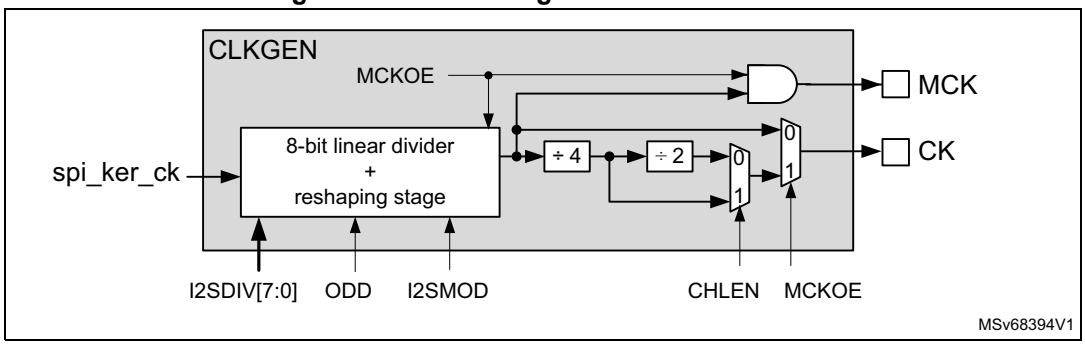


Note: In slave mode, the stop sequence is only controlled by the SPE bit.

55.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the frame synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.
If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 928. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN, and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there are no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: MCK duty cycle is 50 %, even for odd division ratios, when the kernel clock duty cycle is 50 %. When the division ratio is set (I2SDIV = 0), the MCK duty cycle is the same as the duty cycle of the kernel clock.

Table 642 provides examples of clock generator programming for I2S modes.

MCK generation

The master clock MCK is generated when the following conditions are met:

- I2SMOD must be equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set,

Table 642. CLKGEN programming examples for usual I2S frequencies

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
12.288	16	12	0	No	16
12.288	32	6	0		16
12.288	16	6	0		32
12.288	32	3	0		32
49.152	16	16	0		48
49.152	32	8	0		48
49.152	16	8	0		96
49.152	32	4	0		96
49.152	16	4	0		192
49.152	32	2	0		192

Table 642. CLKGEN programming examples for usual I2S frequencies (continued)

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
4.096	16 or 32	0	-	Yes	16
24.576	16 or 32	3	0		
49.152	16 or 32	3	0		32
12.288	16 or 32	0	-		48
49.152	16 or 32	2	0		
61.44	16 or 32	2	1		
98.304	16 or 32	2	0		96
196.608	16 or 32	2	0		192

55.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

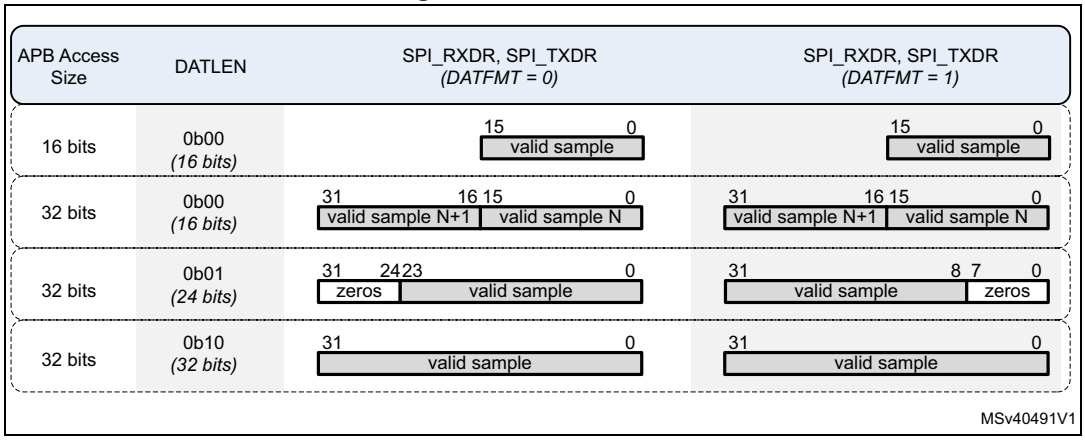
It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 929 shows the allowed settings between APB access sizes, DATFMT, and DATLEN.

Note: Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 929. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 55.3: SPI implementation* to check the supported data size.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to *Section 55.9.11* and *Section 55.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes three basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)[a],
- 4 samples, if the DATLEN = 2 (32 bits)[a].

55.9.11 FIFO status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. Thus, FTHLV new data to be transmitted can be written to SPI_TXDR. The TXP flag is reset when the number of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 55.10* for additional information on the interrupt function in I2S mode.

55.9.12 Handling of underrun situation

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

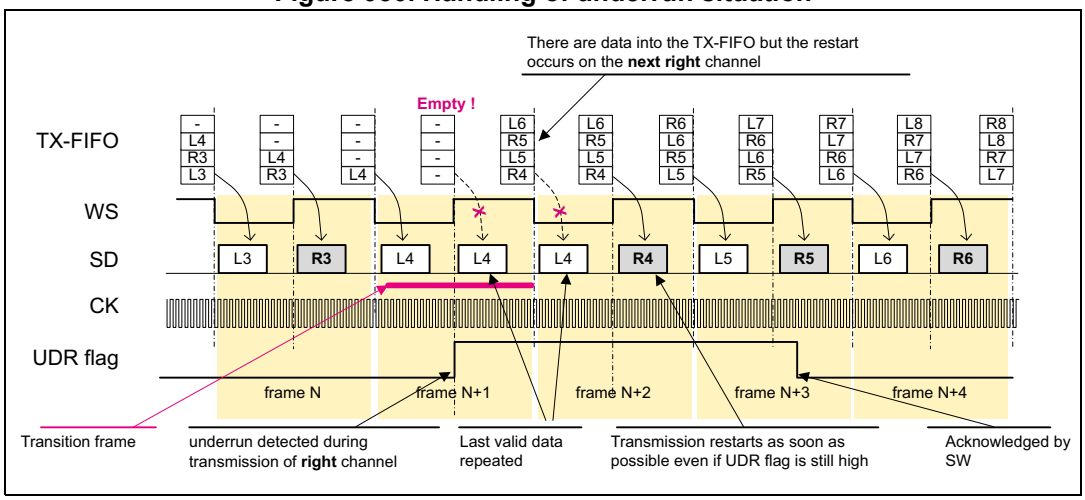
a. Not always available, refer to *Section 55.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral replays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the restart condition is met:
  - If the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - If the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

Figure 930. Handling of underrun situation



When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO.

The UDR flag can trigger an interrupt if the UDRIE bit of the SPI_IER register is set. The UDR bit is cleared by setting the UDRC bit of the SPI_IFCR register.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK, and MCK signals are not gated.

Due to resynchronization, any change of the UDR flag is taken into account by the hardware after at least two periods of CK clock.

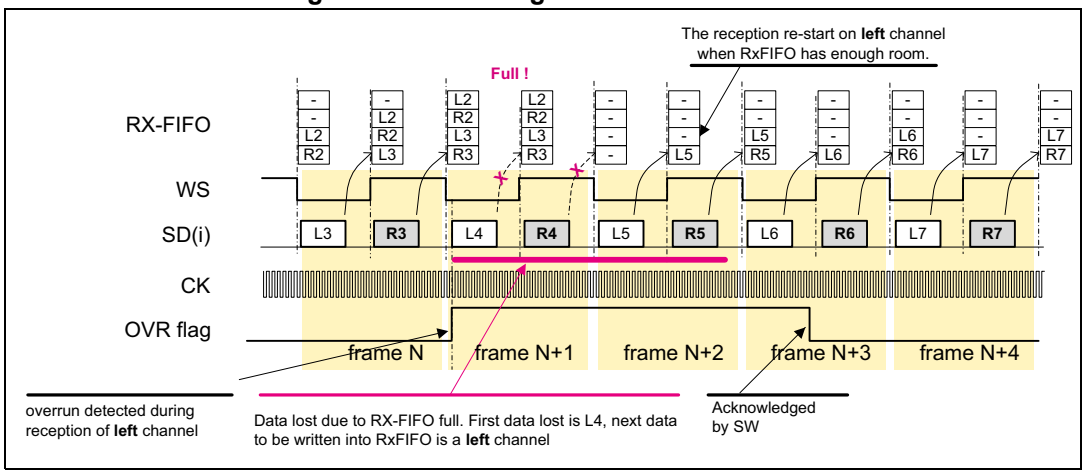
55.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.

In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the RxFIFO. The block starts by writing next the right channel into the RxFIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

Figure 931. Handling of overrun situation



When the block is configured in PCM mode, after an overrun error, the block stops writing data in the RxFIFO as long as conditions of restart are not met. When there is enough room in the RxFIFO, the next received data are written in the RxFIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by setting the OVRC bit of the SPI_IFCR register.

Note: An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK, and MCK signals are not gated.

55.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

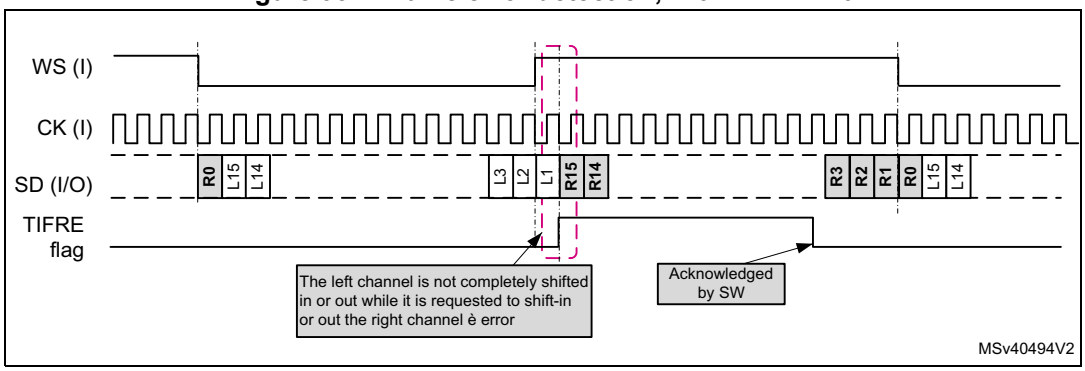
- A frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurs at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.

Figure 932 shows an example of frame error detection. The SPI/I2S block is in slave mode and the number of bit clock periods for the left channel are not enough to shift-in or shift-out the data. The figure shows that the ongoing transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 932. Frame error detection, with FIXCH = 0

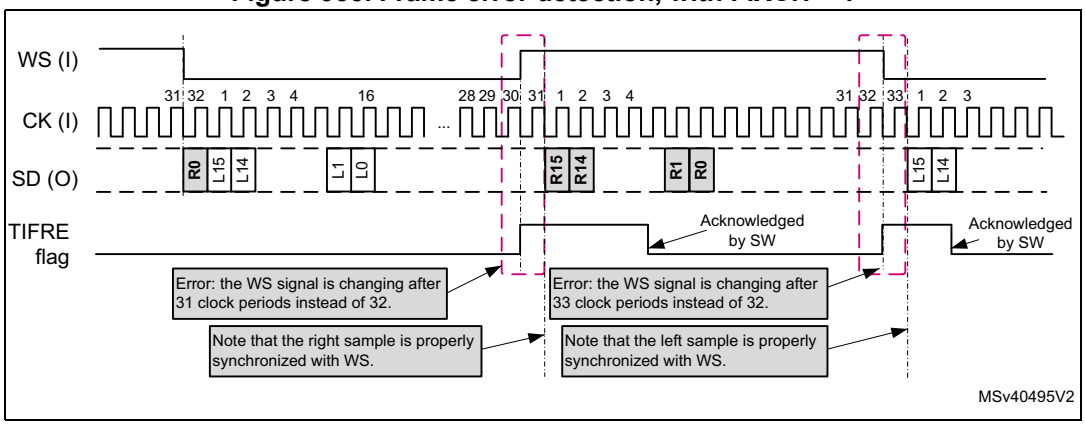


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 933*, in this mode, the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 932* and *Figure 933* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 933. Frame error detection, with FIXCH = 1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: *The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.*

55.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of the SPI_CFG1 register.

In receive mode, the DMA interface is working as follows:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follows:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

55.9.16 Programing examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set I2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example, if a threshold of two audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. If needed, the application can then configure the external audio codec via an I2C interface.
8. Clear all interrupt enable fields located in the SPI_IER register.
9. Clear all status flags, by setting SUSPC, TIFREC, OVRC, UDRC of the SPI_IFCR register.
10. When the application wants to start the data transfer, set the SPE bit. Then:
    - If the data transfer uses DMA:
    a) Program the DMA peripheral: two channels, one for RX and one for TX.
    b) Initialize the memory buffer with valid audio samples for TX path.
    c) Enable the DMA channels.
    d) Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
    e) Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled up.
    - If the data transfer is done via interrupt:

      Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
11. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Stop procedure in master mode

1. Set the bit CSUSP, in order to stop ongoing transfers
2. Check the value of the CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the

SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So, we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example if a threshold of two audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:
   – Program the DMA peripheral: one RX channel
   – Enable the DMA channel,
   – In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.
8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally, the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop procedure in slave mode

1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

55.10 I2S interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wake-up event signal (**spi_wkup**) can be generated according to the events described in the *Table 643*.

Interrupt events can be enabled and disabled separately.

Table 643. I2S interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/Interrupt clearing method	Exit Sleep mode	Exit Stop modes	Exit Standby mode
SPI	TxFIFO threshold reached	TXP	When the TxFIFO contains less than FTHLV empty locations	Yes	Yes	No
	RxFIFO threshold reached	RXP	When the RxFIFO contains less than FTHLV samples			
	Overrun error	OVR	Write OVRC to 1			
	Underrun error	UDR	Write UDRC to 1			
	Frame error flag	TIFRE	Write TIFREC to 1		No	

55.11 SPI/I2S registers

55.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

0: AF configuration is not locked

1: AF configuration is locked

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: all zero pattern is applied

1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: All zero pattern is applied

1: All ones pattern is applied

Bit 13 **CRC33_17**: Full size (33-bit or 17-bit) CRC polynomial configuration

0: Full size (33-bit or 17-bit) CRC polynomial is not used

1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** Internal slave select signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral NSS input internally and the I/O value of the NSS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at half-duplex mode

In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transfer.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer starts.

Bit 9 **CSTART:** master transfer start

This bit can be set by software if SPI is enabled only to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the *Section 55.9.8: Stop sequence*.

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why the automatic suspension feature is not reliable under one or both of the following conditions, where the master may still enter an overrun state and lose some received data:

– When data frames are configured to be shorter than 8 bits.

– When the SCK pad is exposed to a high external capacitive load.

This can cause a significant delay in synchronizing the internal SCK feedback signal, resulting in master-suspension latency that exceeds a single SCK bit period.

To prevent any overrun state, it is strongly recommended to set the MIDI parameter to a value equal to or greater than 4.

After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transfer without any next constraint. Before the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the NSS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero. SPE is cleared and cannot be set when MODF error flag is active.

0: Serial peripheral disabled.

1: Serial peripheral enabled

55.11.2 SPI/I2S control register 2 (SPI_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0xFFFF for a full-featured instance and 0x3FF for a limited-featured instance, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

55.11.3 SPI/I2S configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	DRDS	Res.	CRCEN	Res.	CRCSIZE[4:0]				
rw	rw	rw	rw				rw		rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator
0: bypass is disabled
1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting
000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256
Note: MBR setting is considered at slave working at TI mode, too (see Section 55.5.1: TI mode).

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **DRDS:** data sampling delay on master input (MISO)
This delay is used to compensate the delay on the MISO line (for example due to the signal optical isolation).
0: Sampling is performed on the configured SCK edge (no delay is applied)
1: Sampling is postponed by a fixed delay equal to a half SCK cycle (the sampling is performed just at the end of the bit validity interval at MISO line).
Note: The DRDS setting has no impact on the other SCK management nor on the data frame communication format. When the delay is applied together with CRC enabled, the CRC computation and evaluation is also delayed.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable
0: CRC calculation disabled
1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transferred and compared

Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.

The value must be equal or a multiple of the data size configured through the DSIZE bitfield. Its maximum size corresponds to the instance maximum data size.

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit at CRCSIZE bit field is reserved for the peripheral instances where data size is limited to 16 bits.

The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

Bit 15 **TXDMAEN:** Tx DMA stream enable

0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable

0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition

For more details see *Figure 914: Optional configurations of the slave behavior when an underrun condition is detected*.

0: slave sends a constant pattern defined by the user in the SPI_UDRDR register

1: Slave repeats the last received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

–If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV = 2, 4, 6.

–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8 bits, better to select FTHLV = 4, 8, 12.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features.

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

55.11.4 SPI/I2S configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE = 0 only.

When SPI must be disabled temporary for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Note: This bit can be also used in PCM and I2S modes.

The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first
Note: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER:** SPI master
0: SPI slave
1: SPI master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
0: no swap
1: MOSI and MISO are swapped
Note: This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE in the SPI_CFG1 register is configured shorter than 8 bits, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: Master Inter-Data Idleness

Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.

0000: no delay

0001: 1 clock cycle period delay

...

1111: 15 clock cycle periods delay

Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

55.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MODFIE**: mode Fault interrupt enable

0: MODF interrupt disabled

1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable

0: TIFRE interrupt disabled

1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable

0: CRC interrupt disabled

1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable

0: OVR interrupt disabled

1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTF interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

55.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus .
Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data
Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE = 0 and data size is set up to 16 bits, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16 bits, these bits are always read as 00. As a result, the single data frame received in the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if the data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Possible value when data size is set up to 8 bits only.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

This flag is set when SPI is reset or disabled.

0: current data transfer is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In master mode, SUSP is set by hardware either as soon as the current frame is complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (master mode active or other mode).

1: master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is disabled or by writing 1 to the CRCEC bit of the SPI_IFCR.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is disabled or by writing 1 to the OVRC bit of the SPI_IFCR.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is disabled or by writing 1 to the UDRC bit of the SPI_IFCR.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only. In I2S/PCM mode, (when available) this flag applies to master and slave mode.

Bit 4 **TXTF:** transmission transfer filled

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and respective DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

55.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear

Writing a 1 into this bit clears SUSP flag of the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear

Writing a 1 into this bit clears MODF flag of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear

Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

55.11.8 SPI/I2S transmit data register (SPI_TXDR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXDR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **TXDR[31:0]:** transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

55.11.9 SPI/I2S receive data register (SPI_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

55.11.10 SPI/I2S polynomial register (SPI_CRCPOLY)

Address offset: 0x040

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCPOLY[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

55.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.
The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of the CRCSIZE bitfield is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

55.11.12 SPI/I2S receiver CRC register (SPI_RXCRC)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.
The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: RXCRC[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

55.11.13 SPI/I2S underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDRDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDRDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

55.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x050

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit, which needs to be kept at 0.

This register is reserved for instances not supporting I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MCKOE**: master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler

Refer to *Section 55.9.9: Clock generator* for details

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler
Refer to *Section 55.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format
0: The data inside the SPI_RXDR or SPI_TXDR register are right aligned
1: The data inside the SPI_RXDR or SPI_TXDR register are left aligned.

Bit 13 **WSINV**: word select inversion
This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB- or LSB-justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB- or LSB-justified mode, the left channel is transfered when WS is LOW, and right channel when WS is HIGH.
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN must be set)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 55.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I²S standard selection
For more details on I²S standards, refer to *Section 55.9.5: Supported audio protocols*
00: I²S Philips standard.
01: MSB-justified standard (left justified)
10: LSB-justified standard (right justified)
11: PCM standard

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - Full Duplex
101: master - Full Duplex
others: not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

55.11.15 SPI/I2S register map

Table 644. SPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	DRDS	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0				0		0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0						0	0	0	0	0	0	0	0
0x010	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x014	SPI_SR	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x018	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x01C	Reserved	Res.																															
0x020	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 644. SPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024-0x02C	Reserved	Res.																															
0x030	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034 - 0x03C	Reserved	Res.																															
0x040	SPI_CRCPOLY	CRCPOLY[31:16](2)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x044	SPI_TXCRC	TXCRC[31:16](2)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SPI_RXCRC	RXCRC[31:16](2)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SPI_UDRDR	UDRDR[31:16](2)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SPI_I2SCFGR	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	Res.	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	Reset value							0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0		0	0	0	0	0	0

1. The configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 55.3: SPI implementation*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

56 FD controller area network (FDCAN)

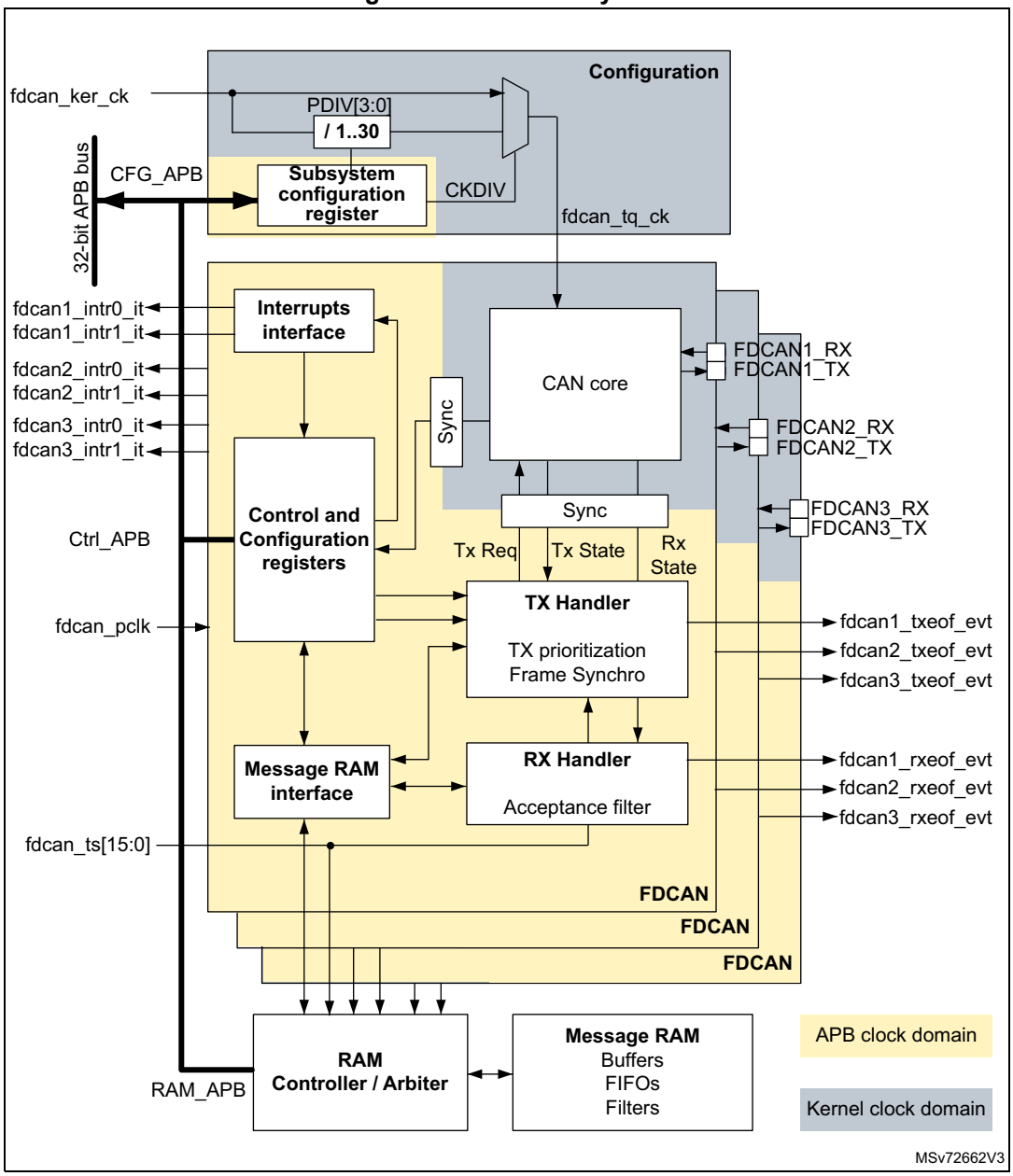
56.1 FDCAN introduction

The controller area network (CAN) subsystem (see *Figure 934*) consists of three CAN modules, a shared message RAM, and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance is used for filtering, transmitting event FIFOs, and receiving and transmitting FIFOs.

Figure 934. CAN subsystem



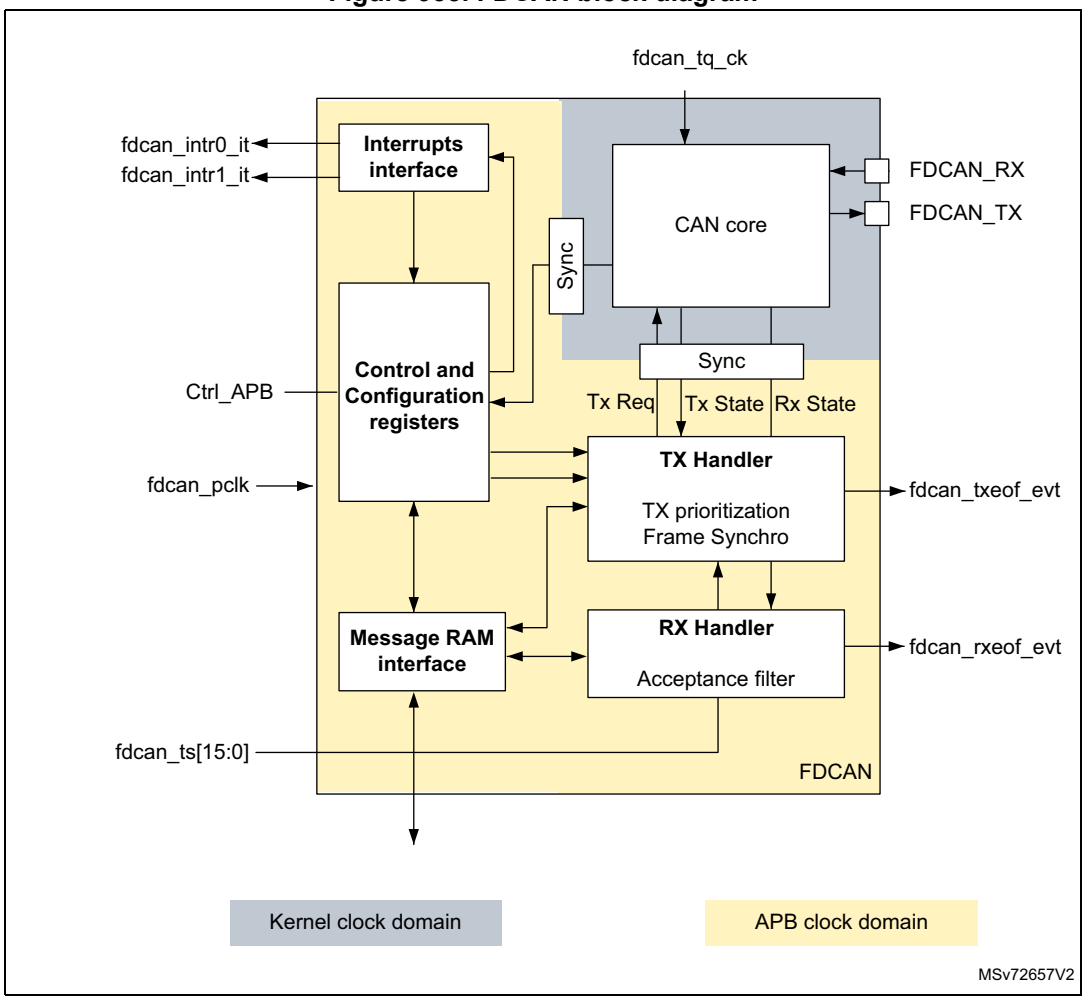
56.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable transmit FIFO/queue of three payloads (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

56.3 FDCAN functional description

56.3.1 FDCAN block diagram

Figure 935. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it.

By programming the EINT0 and EINT1 bits of the FDCAN_ILE register, the interrupt lines can be independently enabled or disabled.

CAN core

The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

Tx handler

The Tx handler controls the message transfer from the message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. The Tx buffer can be used as Tx FIFO or as a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

The Tx handler also generates an event on the fdcan_txeof_evt output when the end-of-frame transmission is issued. This event is internally connected to the TIM2 and TIM3 input capture.

Rx handler

The Rx handler controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs; up to eight filters for 29-bit IDs.

The Rx handler also generates an event on the fdcan_rxeof_evt output when the end-of-frame reception is received. This event is internally connected to the TIM2 and TIM3 input capture.

APB interface

The APB interface connects the FDCAN to the APB bus for configuration registers, controller configuration, and RAM access.

Message RAM interface

The message RAM interface connects the FDCAN access to an external 1-Kbyte message RAM through a RAM controller/arbiter.

56.3.2 FDCAN pins and internal signals

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 645*, *Table 646*, and *Figure 934*.

Table 645. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan*n*_intr0_it	Digital output	FDCAN interrupt0
fdcan*n*_intr1_it		FDCAN interrupt1
fdcan_ts[15:0]	-	External timestamp vector
fdcan*n*_txeof_evt	Digital output	End-of-frame transmission event
fdcan*n*_rxeof_evt	Digital output	End-of-frame reception event
APB interface	Digital input/output	Single APP with multiple psel for configuration, control and RAM access

Table 646. CAN subsystem I/O pins

Name	Type	Description
FDCAN*n*_RX	Digital input	FDCAN receive pin
FDCAN*n*_TX	Digital output	FDCAN transmit pin

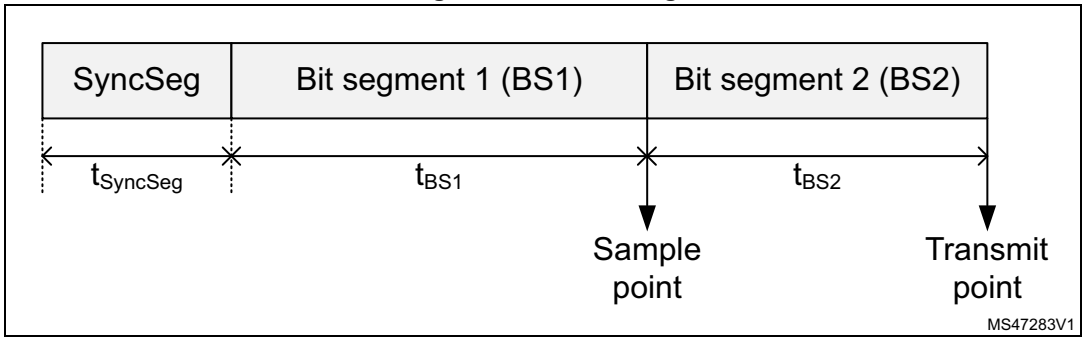
56.3.3 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in *Figure 936*, this operation can be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum ($1 \times t_q$).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable from 1 to 32 time quanta, but can be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but can also be automatically shortened to compensate for negative phase drifts.

Figure 936. Bit timing



The baud rate is the inverse of the bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components (see *Figure 936*):

bit time = $t_{SyncSeg} + t_{BS1} + t_{BS2}$

Where:

– For the nominal bit time

$t_q = (NBRP[8:0] + 1) \times t_{fdcan_tq_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (NTSEG1[7:0] + 1)$

$t_{BS2} = t_q \times (NTSEG2[6:0] + 1)$

Where NBRP[8:0], NTSEG1[7:0], and NTSEG2[6:0] bitfields belong to the FDCAN_NBTP register.

– For the data bit time
$t_q = (DBRP[4:0] + 1) \times t_{fdcan_tq_clk}$
$t_{SyncSeg} = 1 \times t_q$
$t_{BS1} = t_q \times (DTSEG1[4:0] + 1)$
$t_{BS2} = t_q \times (DTSEG2[3:0] + 1)$
Where DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] belong to the FDCAN_DBTP register.

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from recessive to dominant bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW, so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. The FDCAN_DBTP and FDCAN_NBTP registers (dedicated, respectively, to data and nominal bit timing) are only accessible when the CCE and INIT of the FDCA_CCCR register are set.

The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock ($f_{dcan_tq_ck} \leq f_{dcan_pclk}$).

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

56.3.4 Operating modes

Configuration

Access to peripheral version, hardware, and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

The reachable FDCAN bitrates are:

- Nominal bitrate: up to 1 Mbit/s
- Data bitrate: between nominal bitrate and 8 Mbit/s

Software initialization

Software initialization is started by setting the INIT bit of the FDCAN_CCCR register, by software, by a hardware reset, or by entering bus-off state. While the INIT bit is set, message transfers from and to the CAN bus are stopped, and the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting the INIT bit does not change any configuration register. Clearing INIT bit of FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (bus-idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when the INIT bit and the CCE bit of the FDCAN_CCCR register are both set.

The CCE bit of the FDCAN_CCCR register can only be set/cleared while the INIT bit of FDCAN_CCCR is set. The CCE bit is automatically cleared when the INIT bit is cleared.

The following registers are reset when the CCE bit of the FDCAN_CCCR register is set:

- FDCAN_HPMS: High priority message status
- FDCAN_RXF0S: Rx FIFO 0 status
- FDCAN_RXF1S: Rx FIFO 1 status
- FDCAN_TXFQS: Tx FIFO/queue status
- FDCAN_TXBRP: Tx buffer request pending
- FDCAN_TXBTO: Tx buffer transmission occurred
- FDCAN_TXBCF: Tx buffer cancellation finished
- FDCAN_TXEFS: Tx event FIFO status

The timeout counter value (TOC[15:0] bit of the FDCAN_TOCV register) is preset to the value configured by the TOP[15:0] of the FDCAN_TOCC register when the CCE bit of the FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit is set.

The following registers can be written only when the CCE bit is cleared:

- FDCAN_TXBAR: Tx buffer add request
- FDCAN_TXBCR: Tx buffer cancellation request

The TEST and the MON bits of the FDCAN_CCCR register can be set only by software while the INIT and the CCE bits of the FDCAN_CCCR register are both set. Both bits can be reset at any time. The DAR bit of FDCAN_CCCR can only be set/cleared while the INIT and CCE bits are both set.

Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT operation mode is not supported.

Once the FDCAN is initialized and the INIT bit of the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, the Tx FIFO or the Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol:

- Long frame mode (LFM), where the data field of a CAN frame may be longer than eight bytes.
- Fast frame mode (FFM), where the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame.

The fast frame mode can be used in combination with the long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF (res and BRS) decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting the PXE bit of the FDCAN_PSR register. When protocol exception handling is enabled (PXHD = 0 in FDCAN_CCCR), this causes the operation state to change from receiver (ACT[1:0] = 10 in FDCAN_PSR) to integrating (ACT[1:0] = 00 in FDCAN_PSR) at the next sample point. If protocol exception handling is disabled (PXHD = 1 in FDCAN_CCCR), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming the FDOE bit of the FDCAN_CCCR register. In case FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx buffer element. With FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx buffer element is set. The FDOE and BRSE bits of the FDCAN_CCCR register can only be changed while the INIT and CCE bits are both set.

With FDOE = 0, the setting of the FDF and BRS bits is ignored, and frames are transmitted in classic CAN format. With FDOE = 1 and BRSE = 0, only the FDF bit of a Tx buffer element is evaluated. With FDOE = 1 and BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with FDF and BRS bits set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes transmit classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non-CAN FD nodes are held in silent mode until programming is complete. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from that of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 647*.

Table 647. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the FDCAN_DBTP register. In the following FDCAN data phase, the fast CAN bit timing is used as defined by the FDCAN_DBTP register. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (t_q), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames, the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of an FDCAN transmission, only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX, the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), and a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of an FDCAN frame is limited by the transceiver loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay. This is enabled by setting the TDC bit of the FDCAN_DBTP register, and described in detail in the ISO11898-1 specification.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCO[6:0] of FDCAN_TDCR. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mt_q (minimum time quantum, one period of fdcan_tq_ck clock).

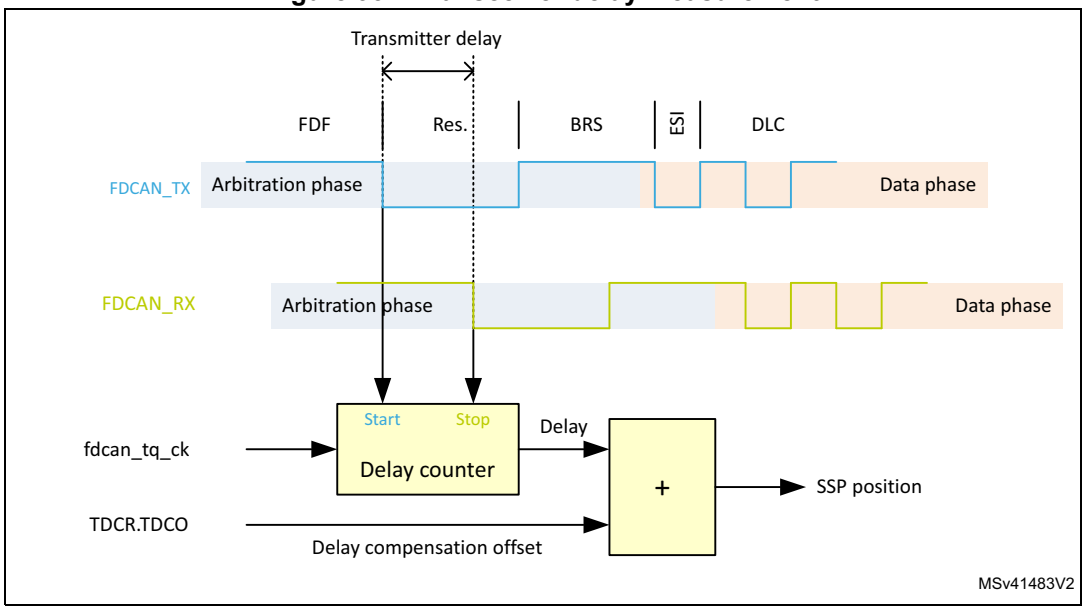
The TDCV[6:0] bitfield of the FDCAN_PSR register shows the actual transmitter delay compensation value. TDCV[6:0] is cleared when the INIT is set in the FDCAN_CCCR. It is updated at each transmission of an FD frame while the TDC bit of the FDCAN_DBTP register is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than 6-bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than or equal to 127 × mt_q. If the sum exceeds this value, the maximum value (127 × mt_q) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by setting the TDC bit of the FDCAN_DBTP; the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mt_q.

Figure 937. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming the TDCF[6:0] bitfield of the FDCAN_TDCR register. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF[6:0] and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode, the node is able to receive data and remote frames, and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits. Instead, it waits for the occurrence of a bus-idle condition to resynchronize itself to the CAN communication. The error counters (REC[6:0] and TEC[7:0] in FDCAN_ECR) are frozen while the error logging (CEL[7:0]) is active. The software can set the FDCAN into restricted operation mode by setting the ASM bit of FDCAN_CCCR. The bit can only be set by software when both CCE and INIT bits are set in FDCAN_CCCR. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler is not able to read data from the message RAM in time. To leave restricted operation mode, the software has to clear the ASM bit of FDCAN_CCCR.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case, the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

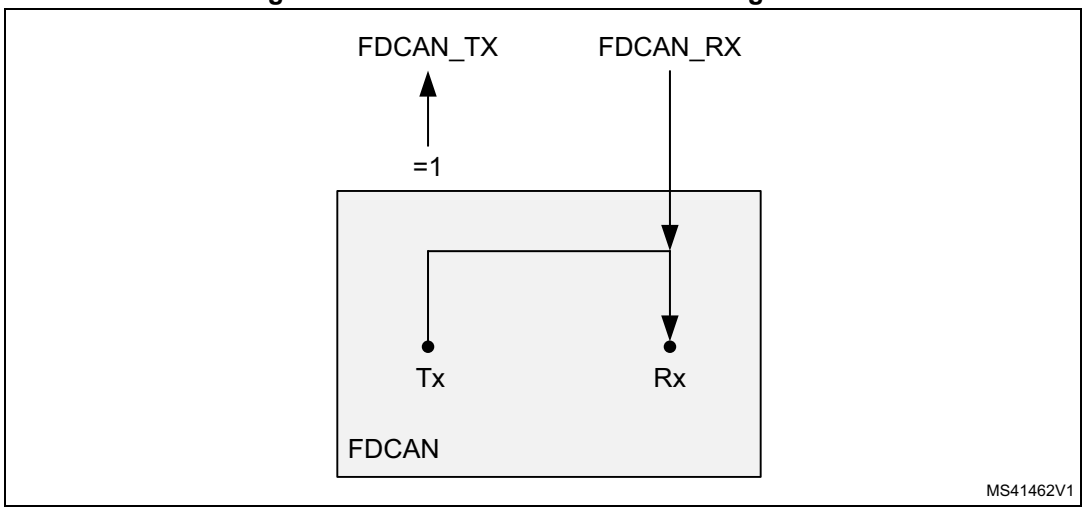
Note: The restricted operation mode must not be combined with the loop-back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting the MON bit of the FDCAN_CCCR register. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In bus monitoring mode, the FDCAN_TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 938* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 938. Pin control in bus monitoring mode



Disabled automatic retransmission mode (DAR)

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled. The DAR mode can be disabled through the DAR bit of the FDCAN_CCCR register.

Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx request pending bit (TRPx in FDCAN_TXBRP) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, when it has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - The corresponding Tx buffer transmission occurred bit TOx is set in FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is cleared in the FDCAN_TXBCF register.
- Successful transmission in spite of cancellation
  - The corresponding Tx buffer transmission occurred bit TOx is set in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.
- Arbitration loss or frame transmission disturbed
  - The corresponding Tx buffer transmission occurred bit TOx is cleared in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.

In case of a successful frame transmission, and if the storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power-down (Sleep mode)

Power-down entry

The FDCAN can be set into power-down mode controlled by setting the CSR bit of the FDCAN_CCCR register. As long as the clock stop request is active, CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until the bus-idle state is detected. The FDCAN then sets the INIT bit of the FDCAN_CCCR register to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power-down by setting the CSA bit of the FDCAN_CCCR register. In this state, before the clocks are switched off, further register accesses can be made. A write access to the INIT bit has no effect. Now, the module clock inputs can be switched off.

Power-down exit

To leave power-down mode, the application has to turn on the module clocks before clearing the CSR bit. The FDCAN acknowledges this by clearing the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

Test modes

To enable write access to *FDCAN test register (FDCAN_TEST)*, the TEST bit of the FDCAN_CCCR register must be set, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming the TX[1:0] bitfield of the FDCAN_TEST register. In addition to its default function (the serial data output), it can drive the CAN sample point signal to monitor the FDCAN bit timing as well as drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from the RX bit of FDCAN_TEST. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there can be a delay of several APB clock periods between writing to TX[1:0] until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via RX.

Note: Test modes must be used for production tests or self-test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop-back mode

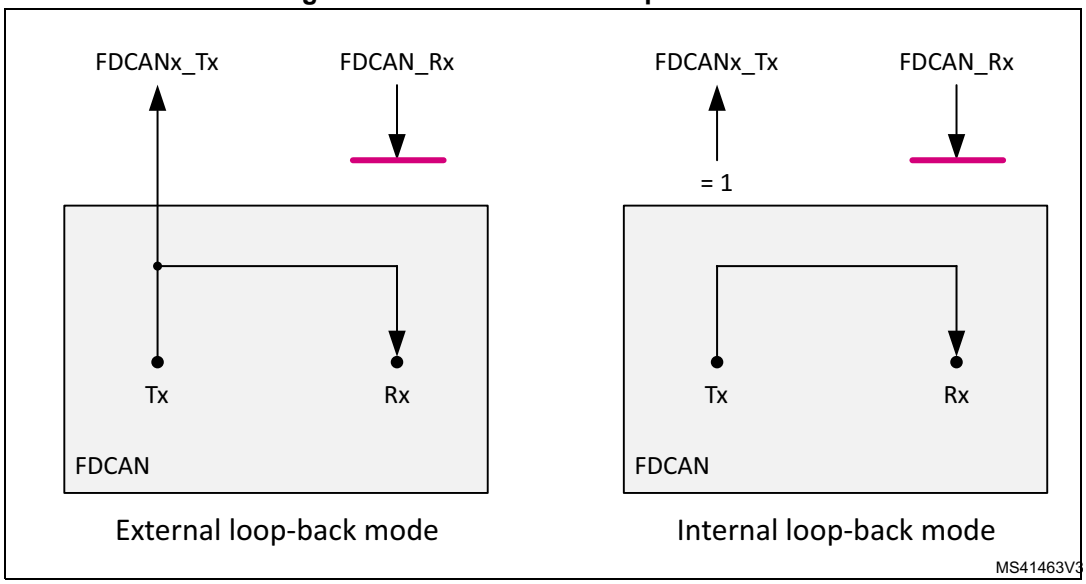
The FDCAN can be set in external loop-back mode by setting the LBCK bit of the FDCAN_TEST register. In loop-back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 939* shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop-back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop-back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop-back mode

Internal loop-back mode is entered by setting both the LBCK bit of FDCAN_TEST and the MON bit of FDCAN_CCR. This mode can be used for a “hot self-test”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. *Figure 939* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop-back mode.

Figure 939. Pin control in loop-back mode



Timestamp generation

For timestamp generation, the FDCAN supplies a 16-bit wrap-around counter. A prescaler (TCP[3:0] of FDCAN_TSCC) can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via the TCV[15:0] bitfield of the FDCAN_TSCV register. A write access to TSCV15:0] resets the counter to 0. When the timestamp counter wraps around, the interrupt flag (TSW bit of FDCAN_ISR) is set.

On start of frame reception/transmission, the counter value is captured and stored into the timestamp section of an Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming TSS[1:0] of FDCAN_TSCC, a 16-bit timestamp can be used.

Debug mode behavior

In debug mode, the set/reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP[3:0] of FDCAN_TSCC as the timestamp counter. The timeout counter is configured via the FDCAN_TOCC register. The actual counter value can be read from the TOC[15:0] bitfield of FDCAN_TOCV. The timeout counter can only be started while the INIT bit of FDCAN_CCCR is cleared. It is stopped when INIT is set, for example, when the FDCAN enters bus-off state.

The operation mode is selected by TOS[1:0] of FDCAN_TOCC. When operating in continuous mode, the counter starts when INIT is cleared. A write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the TOO interrupt flag is set in the FDCAN_IR register. In continuous mode, the counter is immediately restarted at TOP[15:0].

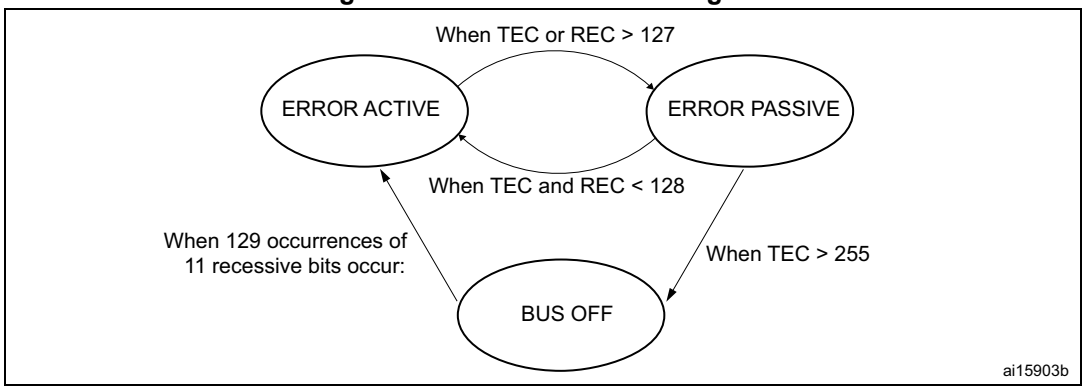
Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization/resynchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in the arbitration and data fields.

56.3.5 Error management

As described in the CAN protocol, the error management is handled entirely by hardware using the transmit error counter (the TEC[7:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*) and the receive error counter (the REC[6:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*). These values are incremented or decremented according to the error condition. For detailed information on TEC[7:0] and REC[6:0] management, refer to the CAN standard. Both values can be read by software to determine the stability of the network.

The bus-off state is reached when TEC[7:0] is greater than 255. This state is also indicated by the BO flag of the *FDCAN protocol status register (FDCAN_PSR)*. In bus-off state, the CAN is no longer able to transmit and receive messages. It has to wait at least for the duration of the recovery sequence specified in the CAN standard (129 occurrences of 11 consecutive recessive bits monitored on FDCAN_RX input).

Figure 940. CAN error state diagram

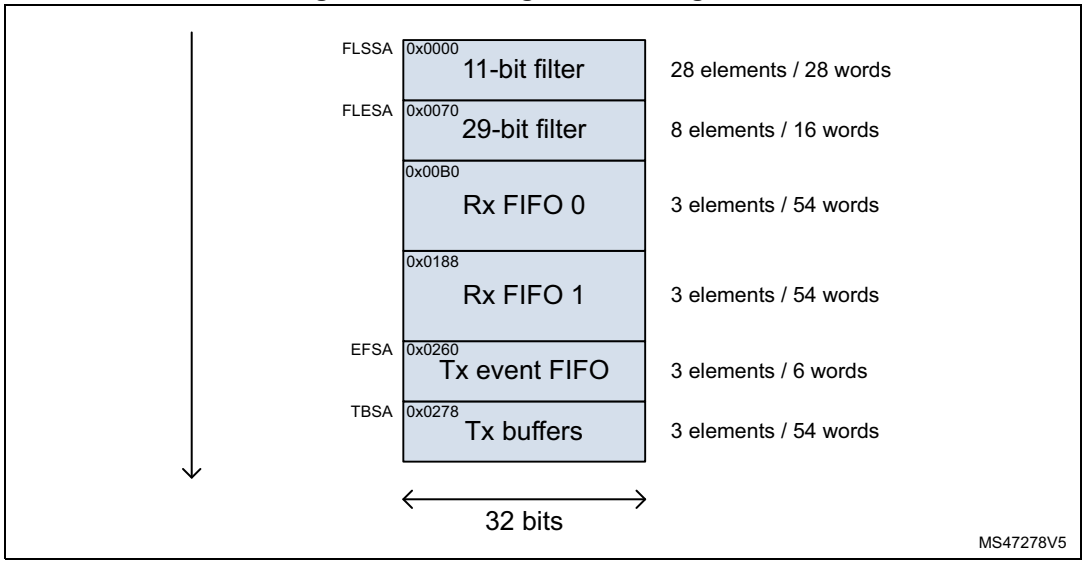


Note: In initialization mode, the CAN does not monitor the FDCAN_RX *signal, and therefore cannot complete the recovery sequence. To recover from an error state, the CAN must operate in normal mode.*

56.3.6 Message RAM

The message RAM is 32-bit wide, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in *Figure 941*.

Figure 941. Message RAM configuration



When the FDCAN addresses the message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, that is, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances, the RAM start address for the FDCANn is computed by end address + 4 of FDCANn - 1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:

- FDCAN1:
  - Start address 0x0000
  - End address 0x034C (as in *Figure 941*)
- FDCAN2:
  - Start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - End address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFO put and get indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element, and the following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled/disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (RXGFC)
- Extended ID AND mask (XIDAM)

Depending on the configuration of the filter element (SFEC[2:0]/EFEC[2:0]), a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set the high priority message interrupt flag HPM in FDCAN_IR
- Set the high priority message interrupt flag HPM in FDCAN_IR, and store the received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (for example, CRC error), this message is discarded with the following impact:

- **Rx FIFO**

  The put index of the matching Rx FIFO is not updated, but the related Rx FIFO element is partly overwritten with the received data. For error type, see LEC[2:0] and DLEC[2:0] bitfields of the FDCAN_PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filters. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT[1:0] = 00: the message ID of received frames is AND-ed with the extended ID AND mask (XIDAM) before the range filter is applied.
- EFT[1:0] = 11: the extended ID AND mask (XIDAM) is not used for range filtering.

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

The classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID/EF1ID is used as message ID filter, while SF2ID/EF2ID is used as filter mask.

0 bit at the filter mask masks out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only the bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

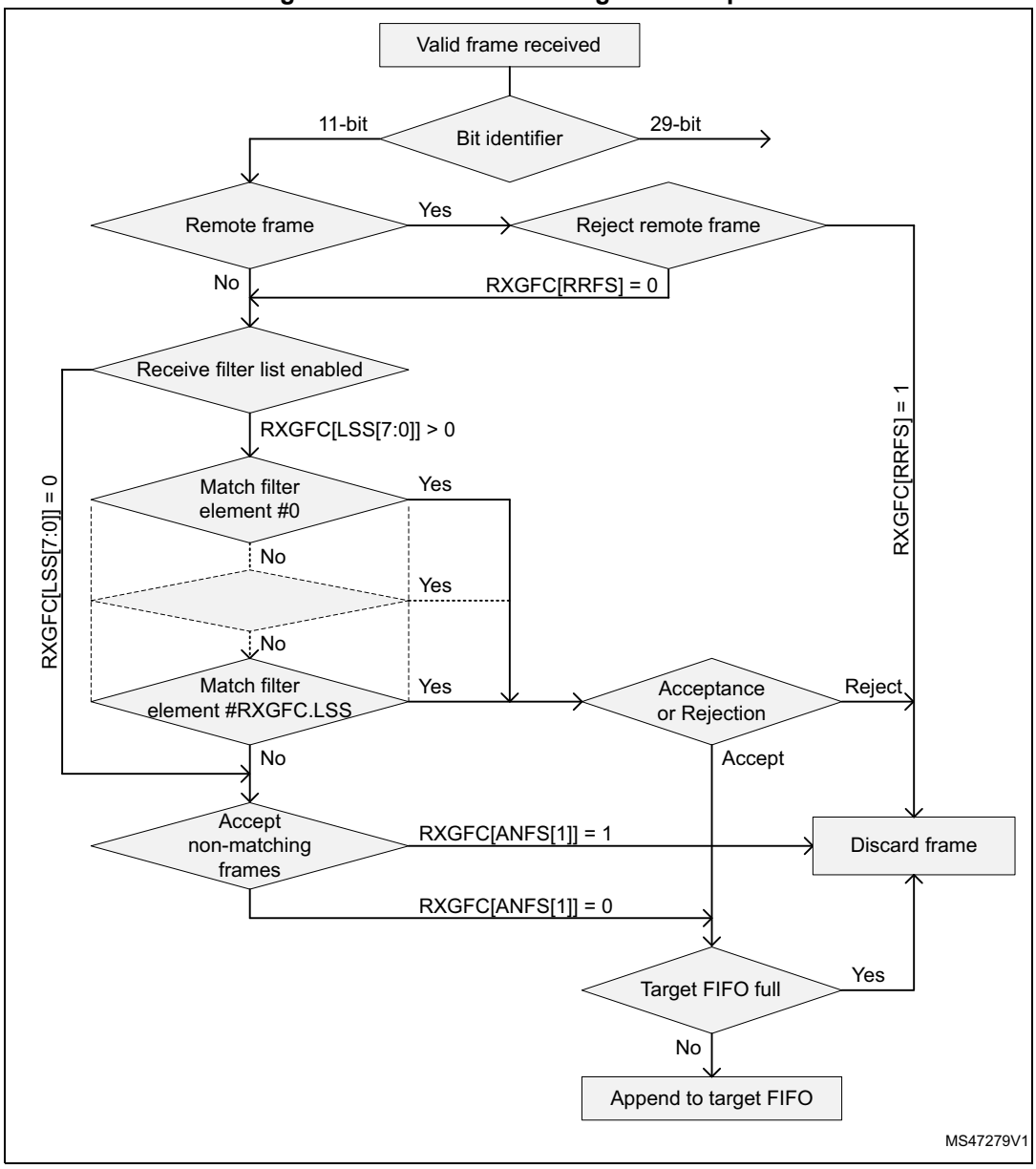
In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 942 shows the flow for standard message ID (11-bit identifier) filtering. The standard message ID filter element is described in *Section 56.3.11*.

The standard message filtering is controlled by the FDCAN_RXGFC register. The standard message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 942. Standard message ID filter path

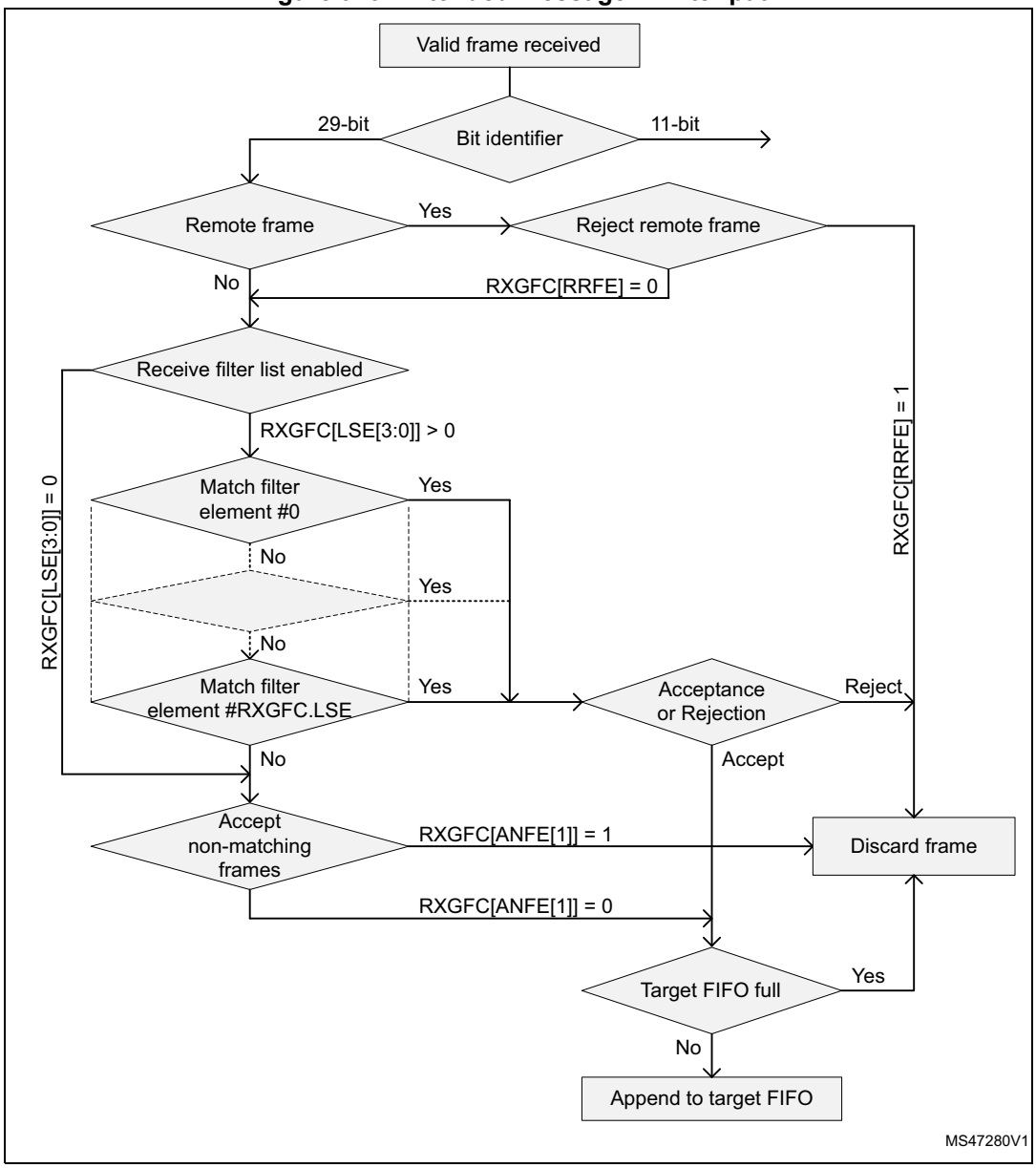


Extended message ID filtering

Figure 943 shows the flow for extended message ID (29-bit identifier) filtering. The extended message ID filter element is described in *Section 56.3.12*.

The extended message filtering is controlled by the FDCAN_RXGFC register. The extended message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 943. Extended message ID filter path



The extended ID AND mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx FIFO element is described in *Section 56.3.8*.

When an Rx FIFO full condition is signaled by RFnF in FDCAN_IR (where n is the FIFO number), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the interrupt flag RFnL is set in the FDCAN_IR register.

When reading from an Rx FIFO, the Rx FIFO get index (FnGI of FDCAN_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address (FnSA).

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by clearing the FnOM bit in the FDCAN_RXGFC register. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (FnPI = FnGI in FDCAN_RXFnS), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by FnF = 1 in FDCAN_RXFnS. In addition, the RFnF interrupt flag is set in FDCAN_IR.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the message lost condition is signaled by setting RFnL bit in FDCAN_RXFnS. In addition, the RFnL interrupt flag is set in FDCAN_IR.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by setting the FnOM bit of the FDCAN_RXGFC register.

When an Rx FIFO full condition (FnPI = FnGI of FDCAN_RXFnS) is signaled by FnF = 1 in FDCAN_RXFnS, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading from the Rx FIFO elements must start at least at get index + 1. This is because it may happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case, inconsistent data can be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index (FnA of FDCAN_RXFnA). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF = 0 in FDCAN_RXFnS).

Tx handling

The Tx handler handles transmission requests for the Tx FIFO and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes. the Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 648. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bit rate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register (FDCAN_TXBRP) is updated, or when a transmission has been started.

Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are permanently specified to specific values and cannot easily be changed. These message identifiers can have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

As an example, if CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus-idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, these messages are started in the idle time, and they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit of the CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, this feature is disabled (TXP = 0 in FDCAN_CCCR).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Tx FIFO

Tx FIFO operation is configured by clearing the TFQM bit of the FDCAN_TXBC register. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index (TFGI[1:0] bitfield of FDCAN_TXFQS). After each transmission, the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order that these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level (TFFL[2:0] bitfield of FDCAN_TXFQS) as the difference between the get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index (TFQPI[1:0] bitfield of FDCAN_TXFQS). An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (TFQF = 1 in FDCAN_TXFQS) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by setting the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via the FDCA_TXBAR register. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with a transmission request is pending and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the message RAM. The Therefore, the start address of the next available (free) Tx FIFO buffer, is calculated by adding 18 times the put index TFQPI[1:0] (0 … 2) to the Tx buffer start address TBSA.

Tx queue

Tx queue operation is configured by setting the TFQM of the FDCAN_TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority).

In case of mixing of standard and extended message IDs, the standard message IDs are compared to bits [28:18] of extended message IDs.

In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index (TFQPI[1:0] in FDCAN_TXFQS). An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (TFQF = 1 in FDCAN_TXFQS), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application can use the FDCAN_TXBRP register instead of the put index and can place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the message RAM. The start address of Therefore, the next available (free) Tx queue buffer is calculated by adding 18 times the Tx queue put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer, the host has to write 1 to the corresponding bit position (= number of Tx buffer) of the FDCAN_TXBCR register. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of the FDCAN_TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN _TXBRP bit remains set as long as the transmission is in progress. If the transmission is successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission is not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This can enable another node to transmit a message that can have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in *Tx FIFO*.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF bit of the FDCAN_IR, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the FDCAN_IR register.

When reading from the Tx event FIFO, the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) has to be added twice to the Tx event FIFO start address EFSA.

56.3.7 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (see *Section 56.4.23* and *Section 56.4.25*). Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value = index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case, the FIFO acknowledge index must not be written because this would set the get index to a wrong position and alter the FIFO fill level. In this case, some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

56.3.8 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in *Table 649*. The description is provided in *Table 650*.

Table 649. Rx FIFO element

Bit	31			24	23							16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]												
R1	ANMF	FIDX[6:0]			Res.			FDF	BRS	DLC[3:0]			RXTS[15:0]			
R2	DB3[7:0]				DB2[7:0]								DB1[7:0]		D[7:0]	
R3	DB7[7:0]				DB6[7:0]								DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮								⋮			
Rn	DBm[7:0]				DBm-1[7:0]								DBm-2[7:0]		DBm-3[7:0]	

The element size configured for storage of CAN FD messages is set to 64-byte data field.

Table 650. Rx FIFO element description

Field	Description
R0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
R0 Bit 30 XTD	Extended identifier Signals to the host whether the received frame has a standard or extended identifier. – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 650. Rx FIFO element description (continued)

Field	Description
R0 Bit 29 RTR	Remote transmission request Signals to the host whether the received frame is a data frame or a remote frame. – 0: Received frame is a data frame – 1: Received frame is a remote frame
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames can be enabled via ANFS[1:0] and ANFE[1:0] bitfield of FDCAN_RXGFC. – 0: Received frame matching filter index FIDX – 1: Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]	Filter index 0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.
R1 Bit 21 FDF	FD format – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit rate switch – 0: Frame received without bit rate switching – 1: Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data length code – 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9-15: Classic CAN: received frame has 8 data bytes – 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.
R2 Bits 31:24 DB3[7:0]	Data byte 3
R2 Bits 23:16 DB2[7:0]	Data byte 2
R2 Bits 15:8 DB1[7:0]	Data byte 1
R2 Bits 7:0 D[7:0]	Data byte 0
R3 Bits 31:24 DB7[7:0]	Data byte 7
R3 Bits 23:16 DB6[7:0]	Data byte 6

Table 650. Rx FIFO element description (continued)

Field	Description
R3 Bits 15:8 DB5[7:0]	Data byte 5
R3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮
Rn Bits 31:24 DBm[7:0]	Data byte m
Rn Bits 23:16 DBm-1[7:0]	Data byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data byte m-3

56.3.9 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration TFQM bit of the FDCAN_TXBC register. The element size is configured for storage of CAN FD messages with up to 64-byte data.

Table 651. Tx buffer and FIFO element

Bit	31			24	23					16	15	8	7	0
T0	ESI	XTD	RTR	ID[28:0]										
T1	MM[7:0]				EFC	Res.	FDF	BRS	DLC[3:0]		Res.			
T2	DB3[7:0]				DB2[7:0]						DB1[7:0]		D[7:0]	
T3	DB7[7:0]				DB6[7:0]						DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮						⋮			
Tn	DBm[7:0]				DBm-1[7:0]						DBm-2[7:0]		DBm-3[7:0]	

Table 652. Tx buffer element description

Field	Description
T0 Bit 31 ESI(1)	Error state indicator – 0: ESI bit in CAN FD format depends only on error passive flag – 1: ESI bit in CAN FD format transmitted recessive
T0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 652. Tx buffer element description (continued)

Field	Description
T0 Bit 29 RTR(2)	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO control – 0: Do not store Tx events – 1: Store Tx events
T1 Bit 21 FDF	FD format – 0: Frame transmitted in classic CAN format – 1: Frame transmitted in CAN FD format
T1 Bit 20 BRS(3)	Bit rate switching – 0: CAN FD frames transmitted without bit rate switching – 1: CAN FD frames transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9 - 15: Classic CAN: received frame has 8 data bytes – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data byte 3
T2 Bits 23:16 DB2[7:0]	Data byte 2
T2 Bits 15:8 DB1[7:0]	Data byte 1
T2 Bits 7:0 D[7:0]	Data byte 0
T3 Bits 31:24 DB7[7:0]	Data byte 7
T3 Bits 23:16 DB6[7:0]	Data byte 6
T3 Bits 15:8 DB5[7:0]	Data byte 5
T3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮

Table 652. Tx buffer element description (continued)

Field	Description
Tn Bits 31:24 DBm[7:0]	Data byte m
Tn Bits 23:16 DBm-1[7:0]	Data byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node can optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if the transmission in CAN FD format is enabled by the FDOE bit of the FDCAN_CCCR.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled by setting the FDOE bit of the FDCAN_CCCR. Bit BRS is only evaluated when in addition BRSE bit is set in FDCAN_CCCR.

56.3.10 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event, FIFO the host CPU gets this information in the order that the messages were transmitted. Status information about the Tx event FIFO can be obtained from FDCAN_TXEFS register.

Table 653. Tx event FIFO element

Bit	31			24	23						16	15		8	7		0
E0	ESI	XTD	RTR	ID[28:0]													
E1	MM[7:0]				ET[1:0]		EDL	BRS	DLC[3:0]			TXTS[15:0]					

Table 654. Tx event FIFO element description

Field	Description
E0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 Bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table 654. Tx event FIFO element description (continued)

Field	Description
E1 Bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
E1 Bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 Bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit rate switching – 0: Frame transmitted without bit rate switching – 1: Frame transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code 0 - 8: Frame with 0-8 data bytes transmitted 9 - 15: Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.

56.3.11 FDCAN standard message ID filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FLSSA plus the index of the filter element (0 … 27).

Table 655. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 656. Standard message ID filter element field description

Field	Description
Bit 31:30 SFT[1:0](1)	Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]	Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
Bits 26:16 SFID1[10:0]	Standard filter ID 1 First ID of standard ID filter element.
Bits 10:0 SFID2[10:0]	Standard filter ID 2 Second ID of standard ID filter element.

1. With SFT[1:0] = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC[2:0] = 000).

Note: In case a reserved value is configured, the filter element is considered disabled.

56.3.12 FDCAN extended message ID filter element

Up to eight filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is the filter list extended start address FLESA plus twice the index of the filter element (0 … 7).

Table 657. Extended message ID filter element

Bit	31		24	23 16	15 8	7 0
F0	EFEC[2:0]		EFID1[28:0]			
F1	EFT[1:0]	Res.	EFID2[28:0]			

Table 658. Extended message ID filter element field description

Field	Description
F0 Bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
F0 Bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism.
F1 Bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bit 29	Not used
F1 Bits 28:0 EFID2[28:0]	Extended filter ID 2 Second ID of extended ID filter element.

56.4 FDCAN registers

56.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: 3

Bits 27:24 **STEP[3:0]**: 2

Bits 23:20 **SUBSTEP[3:0]**: 1

Bits 19:16 **YEAR[3:0]**: 4

Bits 15:8 **MON[7:0]**: 12

Bits 7:0 **DAY[7:0]**: 18

56.4.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

Note: The register read must give the reset value to ensure no endianness issue.

56.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR are set. The CAN time quantum can be programmed in the range of 1 to 32 FDCAN clock periods: t_q = (DBRP[4:0] + 1) FDCAN clock periods.

DTSEG1[4:0] is the sum of PROP_SEG and PHASE_SEG1. DTSEG2[3:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [DTSEG1[4:0] + DTSEG2[3:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
0: Transceiver delay compensation disabled
1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bit rate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values for the baud rate prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is t_{BS1} = (DTSEG1[4:0] + 1) × t_q.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (DTSEG2[3:0] + 1) × t_q.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1: t_{SJW} = (DSJW[3:0] + 1) × t_q.

Note: With an FDCAN clock of 8 MHz, the reset value 0x0000 0A33 configures the FDCAN for a fast bit rate of 500 kbit/s.

The data phase bit rate must be higher than or equal to the nominal bit rate.

56.4.4 FDCAN test register (FDCAN_TEST)

Write access to this register is enabled by setting the TEST bit of the FDCAN_CCCR register. All register functions are set to their reset values when this bit is cleared.

Loop-back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX[1:0] differently from 00 can disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RX**: Receive pin

This bit is used to monitor the actual value of FDCANx_RX. It is synchronized with the FDCANx_RX pin, so it is set after reset if the FDCAN is connected to a network.

0: The CAN bus is dominant (FDCANx_RX = 0)

1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx_TX

10: Dominant (0) level at pin FDCANx_TX

11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop-back mode

0: Reset value, loop-back mode is disabled

1: Loop-back mode is enabled (see *Power-down (Sleep mode)*)

Bits 3:0 Reserved, must be kept at reset value.

56.4.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured through the WDC[7:0] bitfield of the FDCAN_RWD register.

The counter is reloaded with WDC[7:0] when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops, and the interrupt flag WDI is set in the FDCAN_IR register. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.
This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

56.4.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and clearing single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non-ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP:** Transmit pause enable
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant t_q required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bit rate switching
0: Bit rate switching for transmissions disabled
1: Bit rate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, FDCAN_TEST holds reset values
1: Test mode, write access to FDCAN_TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
This bit can only be set by software when both CCE and INIT are set. The bit can be cleared by the host at any time.
0: Bus monitoring mode disabled
1: Bus monitoring mode enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus is idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN can be set in power-down by stopping APB clock and kernel clock.

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or over.load frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus-idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set. The bit can be cleared by the software at any time.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers.
1: The CPU has write access to the protected configuration registers (while INIT set in FDCAN_CCCR).

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization started

Note: Due to the synchronization mechanism between the two clock domains, there can be a delay until the value written to INIT can be read back. Therefore, the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

56.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR register are both set. The CAN bit time can be programed in the range of 4 to 81 × t_q. The CAN time quantum can be programmed in the range of 1 to 1024 FDCAN kernel clock periods:
t_q = (BRP + 1) × FDCAN clock period fdcan_ker_ck.

NTSEG1[7:0] is the sum of PROP_SEG and PHASE_SEG1. NTSEG2[6:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [NTSEG1[7:0] + NTSEG2[6:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 24:16 **NBRP[8:0]**: Bit rate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

This bitfield is write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN kernel clock of 8 MHz, the reset value 0x0600 0A03 configures the FDCAN for a bit rate of 500 kBit/s.

56.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode, the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS[1:0] = 10).

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM2 value (tim2_cnt[15:0])

11: Same as 00.

These bits are write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

56.4.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSS[1:0] = 01 in FDCAN_TSCC, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of TCP[3:0] in FDCAN_TSCC. A wrap around sets the TSW interrupt flag in FDCAN_IR. Write access resets the counter to 0.
When TSS[1:0] = 10, TSC[15:0] reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 that is not caused by write access to FDCAN_TSCV.

56.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored.
00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **ETOC**: Timeout counter enable

0: Timeout counter disabled
1: Timeout counter enabled
This bit is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

For more details, see *Timeout counter*.

56.4.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of the TCP[3:0] bitfield of the FDCAN_TSCC register. When decremented to 0, the TOO interrupt flag is set in FDCAN_IR and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS[1:0] in FDCAN_TOCC.

56.4.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL[7:0]. The counter stops at 0xFF; the next increment of TEC[7:0] or REC[6:0] sets the ELO interrupt flag in FDCAN_IR.

Access type is rc_r: cleared on read.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128.

1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When the ASM bit of the FDCAN_CCCR is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL[7:0] is still incremented.

56.4.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs	rs	rs	r	r	r	r	r	rs	rs	rs

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCO[6:0] in FDCAN_TDCR. The SSP position is, in the data phase, the number of minimum time quanta (mt_q) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 × mt_q.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access

1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.

0: Since this bit was cleared by the CPU, no FDCAN message has been received.

1: Message in FDCAN format with EDL flag set has been received.

Access type is rc_r: cleared on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its BRS flag set.

1: Last received FDCAN message had its BRS flag set.

Access type is rc_r: cleared on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its ESI flag set.

1: Last received FDCAN message had its ESI flag set.

Access type is rc_r: cleared on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC[2:0]. This field is cleared when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.

Access type is rs: set on read.

Bit 7 **BO**: Bus-off status

0: The FDCAN is not in bus-off state.

1: The FDCAN is in bus-off state.

Bit 6 **EW**: Warning status

0: Both error counters are below the error-warning limit of 96.

1: At least one of error counter has reached the error-warning limit of 96.

Bit 5 **EP**: Error passive

0: The FDCAN is in the error-active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The FDCAN is in the error-passive state.

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module's CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication.

01: Idle: node is neither receiver nor transmitter.

10: Receiver: node is operating as receiver.

11: Transmitter: node is operating as transmitter.

Bits 2:0 **LEC[2:0]:** Last error code

LEC[2:0] indicates the type of the last error to occur on the CAN bus. This bitfield is cleared when a message has been transferred (reception or transmission) without error.

000: No error occurred since LEC[2:0] has been cleared by successful reception or transmission.

001: Stuff error. More than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error. A fixed format part of a received frame has the wrong format.

011: Ack error. The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During bus-off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRC error. The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: No change. Any read access to the protocol status register reinitializes LEC[2:0] to 7. When the LEC[2:0] shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is rs: set on read.

Note: When a frame in FDCAN format has reached the data phase with the BRS flag set, the next CAN event (error or valid frame) is shown in DLEC[2:0] instead of LEC[2:0]. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a form error, not as a stuff error.

The bus-off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or clearing the INIT bit of the FDCAN_CCCR register. If the device enters bus-off, it sets the INIT bit of its own, stopping all bus activities. Once INIT has been cleared by the CPU, the device waits for 129 occurrences of bus-idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error management counters are reset. During the waiting time after clearing INIT, each time a sequence of 11 recessive bits has been monitored, a bit0 error code is written to LEC[2:0] of FDCAN_PSR, enabling the CPU to check up whether the CAN bus is

stuck at dominant or continuously disturbed, and to monitor the bus-off recovery sequence. The REC[6:0] bitfield of the FDCAN_ECR register is used to count these sequences.

56.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 × mt_q.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

56.4.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing 1 to the corresponding bit position.

Writing 0 has no effect. A hard reset clears the register. The configuration of FDCAN_IE controls whether an interrupt is generated. The configuration of FDCAN_ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (DLEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (LEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 20 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 19 **BO**: Bus-off status
0: Bus-off status unchanged
1: Bus-off status changed

Bit 18 **EW**: Warning status
0: Error-warning status unchanged
1: Error-warning status changed

Bit 17 **EP**: Error passive
0: Error-passive status unchanged
1: Error-passive status changed

Bit 16 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow.
1: Overflow of CAN error logging counter occurred.

Bit 15 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 14 **MRAF**: Message RAM access failure
The flag is set when the Rx handler:
- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure, the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the host CPU has to clear the ASM of the FDCAN_CCCR register.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 13 **TSW**: Timestamp wraparound
0: No timestamp counter wrap-around
1: Timestamp counter wrapped around

Bit 12 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost

Bit 11 **TEFF**: Tx event FIFO full
0: Tx event FIFO Not full
1: Tx event FIFO full

Bit 10 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element.

Bit 9 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 8 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 7 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 6 **HPM**: High-priority message
0: No high-priority message received
1: High-priority message received

Bit 5 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost

Bit 4 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 3 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 2 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost

Bit 1 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 0 **RF0N**: Rx FIFO 0 new message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

56.4.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARAE**: Access to reserved address enable

Bit 22 **PEDE**: Protocol error in data phase enable

Bit 21 **PEAE**: Protocol error in arbitration phase enable

Bit 20 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **BOE**: Bus-off status
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **HPME**: High-priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

56.4.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific group of interrupt flags from the interrupt register to one of the two module interrupt lines. For interrupt generation, the respective interrupt line must be enabled via the EINT0 and EINT1 bit of the FDCAN_ILE register. 0 indicates that the interrupt is assigned to interrupt line fdcan1_intr0_it, 1 indicates that the interrupt is assigned to interrupt line fdcan1_intr1_it.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERR:** Protocol error grouping the following interruption
- ARAL: Access to reserved address line
- PEDL: Protocol error in data phase line
- PEAL: Protocol error in arbitration phase line
- WDIL: Watchdog interrupt line
- BOL: Bus-off status
- EWL: Warning status interrupt line

Bit 5 **BERR:** Bit and line error grouping the following interruption
- EPL Error passive interrupt line
- ELOL: Error logging overflow interrupt line

Bit 4 **MISC:** Interrupt regrouping the following interruption
- TOOL: Timeout occurred interrupt line
- MRAFL: Message RAM access failure interrupt line
- TSWL: Timestamp wraparound interrupt line

Bit 3 **TFERR:** Tx FIFO ERROR grouping the following interruption
- TEFLL: Tx event FIFO element lost interrupt line
- TEFFL: Tx event FIFO full interrupt line
- TEFNL: Tx event FIFO new entry interrupt line
- TFEL: Tx FIFO empty interrupt line

Bit 2 **SMSG:** Status message bit grouping the following interruption
- TCFL: Transmission cancellation finished interrupt line
- TCL: Transmission completed interrupt line
- HPML: High-priority message interrupt line

Bit 1 **RXFIFO1:** RX FIFO bit grouping the following interruption
- RF1LL: Rx FIFO 1 message lost interrupt line
- RF1FL: Rx FIFO 1 full interrupt line
- RF1NL: Rx FIFO 1 new message interrupt line

Bit 0 **RXFIFO0:** RX FIFO bit grouping the following interruption
RF0LL: Rx FIFO 0 message lost interrupt line
RF0FL: Rx FIFO 0 full interrupt line
RF0NL: Rx FIFO 0 new message interrupt line

56.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming the EINT0 and EINT1 bits.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

56.4.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for message ID filtering. The global filter configuration controls the filter path for standard and extended messages as described in *Figure 942* and *Figure 943*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSE[3:0]				Res.	Res.	Res.	LSS[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
						rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **LSE[3:0]**: Number of extended filter elements in the list
0: No extended message ID filter
1 to 8: Number of extended message ID filter elements
> 8: Values greater than 8 are interpreted as 8.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **LSS[4:0]**: Number of standard filter elements in the list
0: No standard message ID filter
1 to 28: Number of standard message ID filter elements
> 28: Values greater than 28 are interpreted as 28.
This bitfield is write protected (P), which means that write access by the bits is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **F0OM**: FIFO 0 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard
Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard IDs
1: Reject all remote frames with 11-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard IDs
1: Reject all remote frames with 29-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

56.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0084

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID mask
For acceptance filtering of extended frames the extended ID AND mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set, the mask is not active.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

56.4.21 FDCAN high-priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLST	Res.	Res.	FIDX[4:0]					MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
r			r	r	r	r	r	r	r				r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element:
0: Standard filter list
1: Extended filter list

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **FIDX[4:0]**: Filter index
Index of matching filter element.
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.

Bits 7:6 **MSI[1:0]**: Message storage indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

56.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x0090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of the RF0L interrupt flag of the FDCAN_IR register. When RF0L is cleared, this bit is also cleared.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer.
Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0.
Range: 0 to 3.

56.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index

After the host has read a message or a sequence of messages from Rx FIFO 0, it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI[2:0]. This sets the Rx FIFO 0 get index (F0GI[1:0] of FDCAN_RXF0S) to F0AI[2:0] + 1 and updates the FIFO 0 fill level (F0FL[3:0] FDCAN_RXF0S).

56.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message lost

This bit is a copy of the RF1L interrupt flag of the FDCAN_IR register. When RF1L is cleared, this bit is also cleared.

0: No Rx FIFO 1 message lost

1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 24 **F1F**: Rx FIFO 1 full

0: Rx FIFO 1 not full

1: Rx FIFO 1 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index

Rx FIFO 1 write index pointer.

Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F1GI[1:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F1FL[3:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1.
Range: 0 to 3.

56.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F1AI[2:0]**: Rx FIFO 1 acknowledge index
After the host has read a message or a sequence of messages from Rx FIFO 1, it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI[2:0]. This sets the Rx FIFO 1 get index (F1GI[1:0] of FDCAN_RXF1S) to F1AI[2:0] + 1 and updates the FIFO 1 fill level (F1FL[3:0] FDCAN_RXF1S).

56.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:0 Reserved, must be kept at reset value.

56.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the FDCAN_TXBRP register. Therefore, the effect of add/cancellation requests can be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]	
										r				r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
						r	r						r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue full
0: Tx FIFO/queue not full
1: Tx FIFO/queue full

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **TFQPI[1:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 3

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TFGI[1:0]**: Tx FIFO get index
Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC)

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TFFL[2:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC).

56.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TRP[2:0]**: Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via the FDCAN_TXBAR register. The bits are cleared after a requested transmission has completed or has been canceled via the FDCAN_TXBCR register.

After the FDCAN_TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of the FDCAN_TXBRP register. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are directly cleared after the corresponding FDCAN_TXBRP bit has been cleared.

After a cancellation has been requested, a finished cancellation is signaled via the FDCAN_TXBCF in the following cases:

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode, all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately. The corresponding FDCAN_TXBRP bit is cleared.

56.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **AR[2:0]**: Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. When no Tx scan is running, the bits are cleared immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

56.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CR[2:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact.

This enables the host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. The bits remain set until the corresponding FDCAN_TXBRP bit is cleared.

0: No cancellation pending

1: Cancellation pending

56.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TO[2:0]**: Transmission occurred.

Each Tx buffer has its own TO bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

56.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CF[2:0]**: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of the FDCAN_TXBAR register.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

56.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TIE[2:0]**: Transmission interrupt enable

Each Tx buffer has its own TIE bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

56.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CFIE[2:0]**: Cancellation finished interrupt enable.
Each Tx buffer has its own CFIE bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

56.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EFGI[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
						r	r						r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element lost
This bit is a copy of the TEFL interrupt flag of the FDCAN_IR. When TEFL is cleared, this bit is also cleared.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **EFPI[1:0]**: Event FIFO put index
Tx event FIFO write index pointer.
Range: 0 to 3.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **EFGI[1:0]**: Event FIFO get index
Tx event FIFO read index pointer.
Range: 0 to 3.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EFFL[2:0]**: Event FIFO fill level
Number of elements stored in Tx event FIFO.
Range: 0 to 3.

56.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index
After the host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI[1:0]. This sets the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) to EFAI[1:0] + 1 and updates the FIFO 0 fill level (EFFL[2:0] of FDCAN_TXEFS).

56.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

Address offset: 0x0100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PDIV[3:0]**: input clock divider

The CAN kernel clock can be divided prior to be used by the CAN subsystem. The rate must be computed using the divider output clock.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

This bitfield is write-protected (P): which means that write access is possible only when the CCE bit of the FDCAN_CCCR register is set.

Note: The clock divider is common to all FDCAN instances. Only FDCAN1 instance has FDCAN_CKDIV register, which changes clock divider for all instances.

If several FDCAN instances are present, the input clock divider must be modified before configuring the other FDCAN instances.

56.4.38 FDCAN register map

Table 659. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
	Reset value									0	0	0	0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				

Table 659. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003C	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRSRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															
0x0050	FDCAN_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 659. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0058	FDCAN_ILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
	Reset value	0																									0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x0060-0x007C	Reserved	Reserved																															
0x0080	FDCAN _RXGFC	Res.	Res.	Res.	Res.	LSE[3:0].				Res.	Res.	Res.	LSS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0
0x0084	FDCAN _XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0088	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	Res.	Res.	FIDX[4:0]						MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
	Reset value																	0			0	0	0	0	0	0	0	0				0	0	0
0x0090	FDCAN _RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	Res.	F0FL[3:0]		
	Reset value							0	0							0	0							0	0						0	0	0	0
0x0094	FDCAN _RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
	Reset value																														0	0	0
0x0098	FDCAN _RXF1S	Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	Res.	F1FL[3:0]		
	Reset value							0	0							0	0							0	0						0	0	0	0
0x009C	FDCAN _RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
	Reset value																														0	0	0
0x00A0-0x00BC	Reserved	Reserved																															
0x00C0	FDCAN _TXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0																							
0x00C4	FDCAN _TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
	Reset value											0				0	0							0	0						0	1	1

Table 659. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C8	FDCAN_TXBRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP2	TRP1	TRP0
	Reset value																														0	0	0
0x00CC	FDCAN_TXBAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR2	AR1	AR0
	Reset value																														0	0	0
0x00D0	FDCAN_TXBCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR2	CR1	CR0
	Reset value																														0	0	0
0x00D4	FDCAN_TXBTO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO2	TO1	TO0
	Reset value																														0	0	0
0x00D8	FDCAN_TXBCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF2	CF1	CF0
	Reset value																														0	0	0
0x00DC	FDCAN_TXBTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE2	TIE1	TIE0
	Reset value																														0	0	0
0x00E0	FDCAN_TXBCIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE2	CFIE1	CFIE0
	Reset value																														0	0	0
0x00E4	FDCAN_TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EFG[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
	Reset value							0	0							0	0							0	0						0	0	0
0x00E8	FDCAN_TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
	Reset value																															0	0
0x0100	FDCAN_CKDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

57 Position encoder interface (POSENC)

This section only applies to STM32G55x/56x devices.

Note: This chapter only discusses ST implementation specifics and do not include details on master interface, which can be obtained here:

- SICK's Hiperface DSL® Master Integration manual (publicly available)

- on request, from Heidenhain (https://endat.heidenhain.com/contact) for EnDat3

57.1 POSENC introduction

The POSENC peripheral is a multi-protocol digital position encoder master interface.
It includes IP cores from several vendors, and provides an interface for unifying register and memory accesses, as well as interrupt and input/output management.

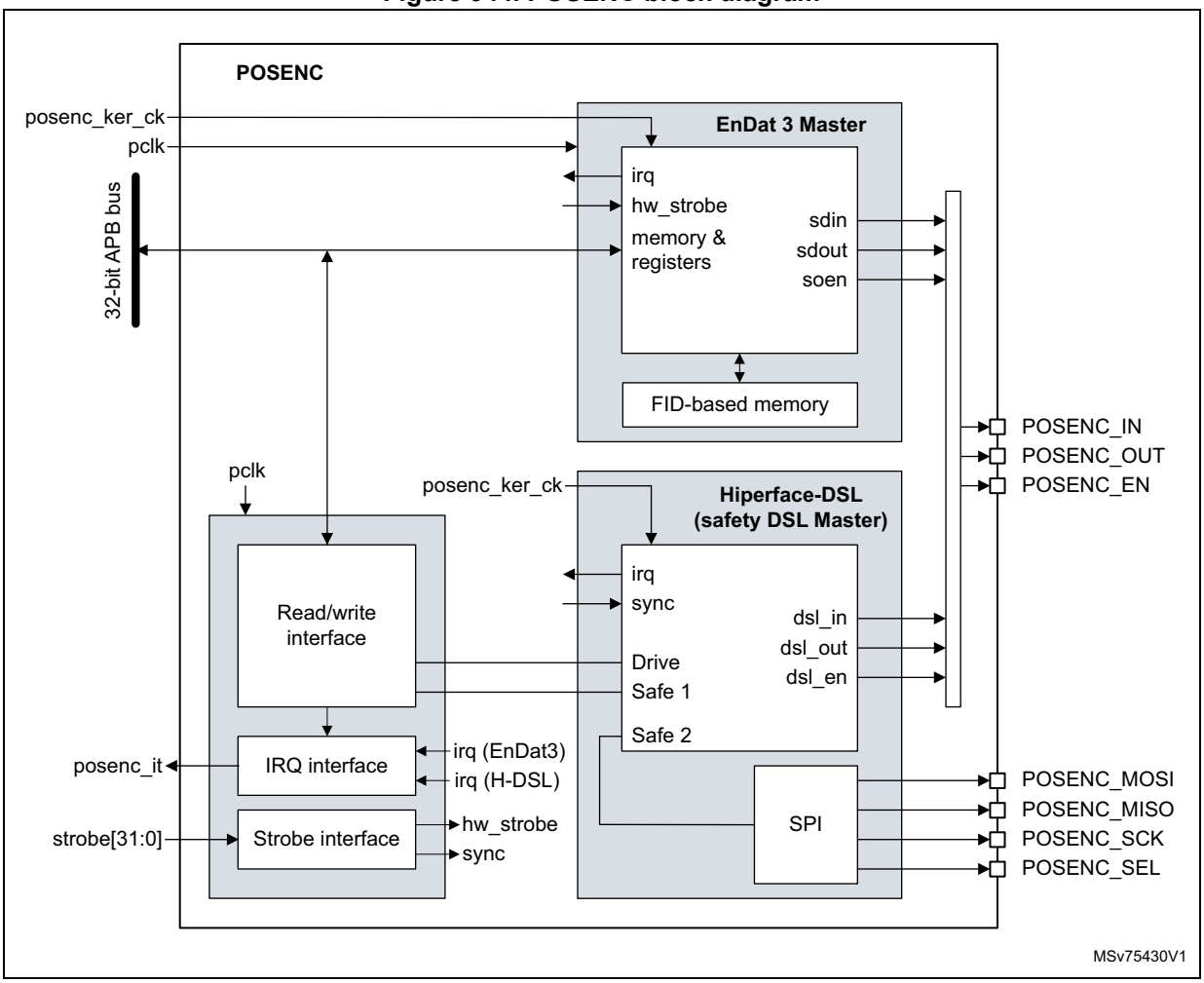
57.2 POSENC main features

- EnDat3 Master
  - HEIDENHAIN IP-core based
  - 12.5 Mbds and 25 Mbds data transmission rate supported
  - Support for FID-based memory
  - Suitable for functional safety (Black channel approach)
  - Support for one safety packet
  - Point-to-point connection (no bus mode supported)
- Hiperface DSL
  - SICK IP-core based
  - safety variants of the DSL Master IP Core
  - Safe channel 2 accessible via SPI-based interface (for external safe control unit)
- Direct or indirect IP cores memory and registers interfaces
- Posted write and indirect read commands mechanisms for minimizing APB bus stall
- Strobe interface for synchronizing communication with on-chip peripherals (e.g timers) or off-chip peripheral or host (via the EXTI external interrupt interface)
- Interrupt interface
- APB interface

57.3 POSENC functional description

57.3.1 POSENC block diagram

Figure 944. POSENC block diagram



57.3.2 POSENC pins and internal signals

Table 660 and *Table 661* summarize the POSENC inputs and outputs.

Table 660. POSENC input/output pins

Pin name	Pin type	Description
POSENC_IN	Input	Input data EnDat3: sdin Hiperface DSL: dsl_in

Table 660. POSENC input/output pins (continued)

Pin name	Pin type	Description
POSENC_OUT	Output	Output data EnDat3: sdout Hiperface DSL:dsl_out
POSENC_EN	Output	Output enable for output data, RS 485 transceiver control EnDat3: soen Hiperface DSL: dsl_en
POSENC_MOSI POSENC_MISO POSENC_SCK POSENC_SEL	Input/Output	(Hiperface DSL only) Serial interface signals: for accessing safe channel 2 (Slave SPI) POSENC_MOSI: SPI data input POSENC_MISO: SPI data output POSENC_SCK: SPI clock POSENC_SEL: SPI selection

Table 661. POSENC internal input/output signals

Signal name	Signal type	Description
posenc_ker_ck	Input	POSENC kernel clock
pclk	Input	POSENC APB clock
posenc_it	Output	POSENC interrupt
strobe[31:0]	Input	strobe input bus. These inputs can be used for synchronizing the position reading with an on-chip or off-chip peripheral.

Table 662. POSENC strobe sources

strobe[31:0]	POSENC1	POSENC2
strobe0	tim1_trgo	
strobe1	tim1_trgo2	
strobe2	tim8_trgo	
strobe3	tim8_trgo2	
strobe4	tim9_trgo	
strobe5	tim12_trgo	
strobe6	tim20_trgo	
strobe7	tim20_trgo2	
strobe8	Reserved	
strobe9	Reserved	
strobe10	tim26_trgo	
strobe11	tim27_trgo	
strobe12	hrtim1_adc_trig1	

Table 662. POSENC strobe sources

strobe[31:0]	POSENC1	POSENC2
strobe13	hrtim1_adc_trig3	
strobe14	Reserved	
strobe15	Reserved	
strobe16	EXTI10	EXTI12
strobe17	play1_out7(1)	play1_out10(1)
strobe[31:18]	Reserved	

1. This connection is only available on STM32G55x/56x devices.

57.3.3 POSENC reset and clocks

The POSENC is a multi-clock domain peripheral. The RCC (reset and clock controller) provides the following clocks to the POSENC:

- APB clock (pclk) used for the register and memory interfaces
- kernel clock (posenc_ker_ck) used by the IP cores

These clocks are not supposed to be phase locked. All signals crossing these clock domains are re-synchronized, except the strobe signals. On these, the synchronization can be disabled when pclk and posenc_ker_ck are derived from exactly the same source (see *Section 57.3.7: POSENC strobe interface*).

Warning: To disable the strobe signal re-synchronization, it is mandatory to have the very same clock source selected for pclk and posenc_ker_ck.

Table 663 summarizes the clock requirements for the protocols.

Table 663. POSENC clock requirements

Protocol	posenc_ker_ck frequency(MHz)	Duty cycle	Tolerance	pclk constraints	IP core inputs
EnDat3	clk_rxtx = 100 MHz or clk_rxtx = 200 MHz	50%	±100 ppm	It is mandatory to have: 48 MHz ≤ pclk ≤ posenc_ker_ck	clk_rxtx = posenc_ker_ck clk_sys = pclk
Hiperface-DSL	clk = 75 MHz	50%	±100 ppm	None	clk = posenc_ker_ck

The POSENC reset signal is propagated simultaneously to all sub-components: common register interface read and write circuitry and IP cores.

57.3.4 POSENC memory organization and accesses

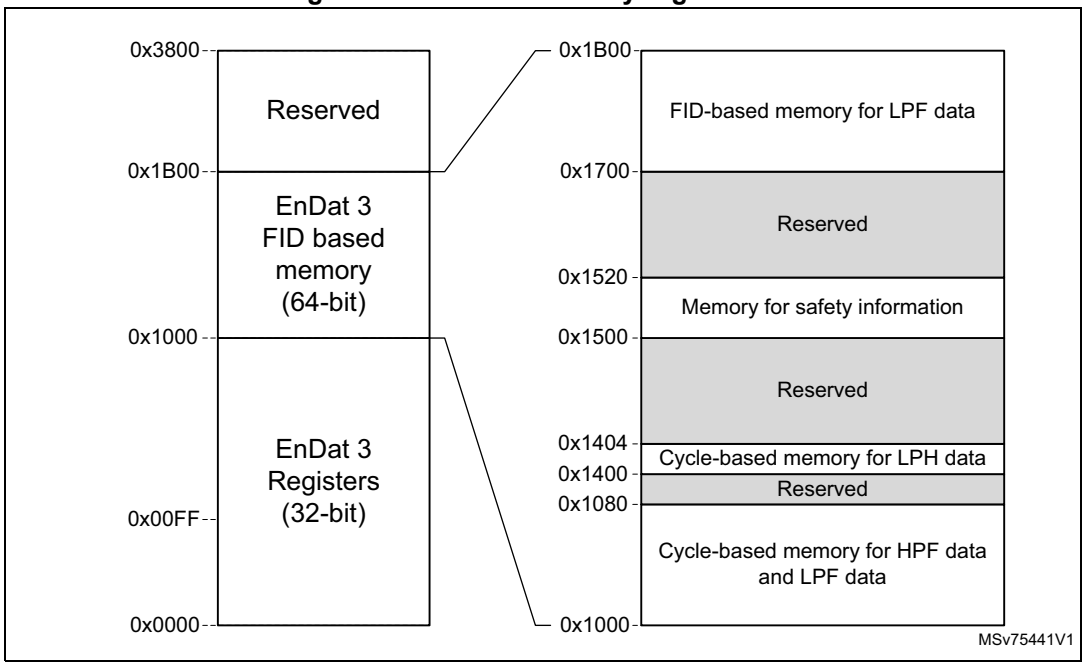
The POSENC memory space is 16 Kbytes, and is divided as follows:

- a lower part (offset 0x0000 to 3xFFF) dedicated to the IP cores accesses, with the possibility to only access one protocol at once, all of them starting from offset 0x0000
- an upper part (0x3800 to 0x3FFF) holding the control and status registers common to all IP cores, such as protocol selection, strobe and interrupt control. This part is accessible whatever the protocol

Figure 945 represents the memory mapping. The memory accesses to the IP cores region is protocol specific and is described in the section below.

Figure 945 details the ENDat3 memory organization.

Figure 945. ENDat3 memory organization



57.3.5 EnDat3 memory accesses

The EnDat3 configuration and status registers (offset 0x0000 to 0x0FFF) are 32 bits, and can be accessed randomly within a cycle APB transfer, read or write.

The accesses to EnDat3 FID based memory (offset 0x1000 to 0x3FFF) must be handled differently, as the memory has a 64-bit width.

The memory must be read by two consecutive 32-bit accesses, and it is mandatory to do the first access with a 64-bit aligned address (such as 0x0000, 0x0008, 0x0010).

Note: Any sequential access, which is not 64-bit aligned (such as 0x0004 followed by 0x0000), gives an unpredictable result.

57.3.6 Hiperface DSL memory accesses

The Hiperface DSL IP core native interface is 8-bit wide, synchronous to the 75 MHz kernel clock. For interfacing with the 32-bit APB bus clocked with pclk, the POSENC features:

- a 8-bit to 32-bit bus width adapter
- an address translation logic
- dedicated read and write mechanisms to avoid stalling the APB bus during the 8-bit accesses

The address translation logic is required either for accessing single 8-bit registers individually, or for gathering several 8-bit registers when convenient.

This reduces the number of accesses on the POSENC interface for multi-byte registers, and provides ready-to-use data for the software (either 16-, 24-, or 32-bit). The consistency of multi-byte data is ensured with the Hiperface-DSL freeze mechanism.This guarantees that no further update is done until all bytes are read and made available on the 32-bit APB bus.

The consistency of 40-bit fast and safe positions information is granted following the sequence described in section *40-bit software-driven read accesses*.

It is also possible to maintain the freeze during multiple 32-bit registers accesses using the HDFRZ bit in POSENC_CR2.

Table 664 below provides the address translation and freeze information for multi-byte registers.

Note: The POSENC_HD_FPOSSTR and POSENC_HD_FPOSMT1 registers do provide the very same information. The POSENC_HD_FPOSSTR is intended to be used with single-turn encoders providing 24-bit position, with the minimum latency (3 bytes accesses only). Using the POSENC_HD_FPOSMT1 register with single-turn encoder will yield the same result with a POS3 byte null value. The same applies for safe position 1 reading (POSENC_HD__VPOSST and POSENC_HD_VPOSMT1 registers).

Table 664. Hiperface-DSL Address conversion table

Name	32-bit address	Hiperface-DSL addresses	Byte 3	Byte 2	Byte 1	Byte 0	Freeze
POSENC_HD_CTRLR	0x0000	0x00 0x01	-	-	SYNC_CTRL	SYS_CTRL	No
POSENC_HD_MASTER_QMR	0x0004	0x03	-	-	-	MASTER_QM	-
POSENC_HD_EVENTR	0x0008	0x04 0x05	-	-	EVENT_H	EVENT_L	No
POSENC_HD_MASKR	0x000C	0x06 0x07 0x08	-	MASK_SUM	MASK_H	MASK_L	No
POSENC_HD_EDGESR	0x0010	0x09	-	-	-	EDGES	-
POSENC_HD_DELAYR	0x0014	0x0A	-	-	-	DELAY	-

Table 664. Hiperface-DSL Address conversion table (continued)

Name	32-bit address	Hiperface-DSL addresses	Byte 3	Byte 2	Byte 1	Byte 0	Freeze
POSENC_HD_VRS_RELR(1)	0x0018	0x0B 0x0C	RELEASE (day)	RELEASE (month)	RELEASE (year)	VERSION	No
POSENC_HD_ENC_IDR	0x001C	0x0D 0x0E 0x0F	-	ENC_ID2	ENC_ID1	ENC_ID0	No
POSENC_HD_FPOSSTR	0x0020	0x12 0x13 0x14	-	POS2	POS1	POS0	Yes
POSENC_HD_FPOSMT1	0x0024	0x11 0x12 0x13 0x14	POS3	POS2	POS1	POS0	Yes
POSENC_HD_FPOSMT2	0x0028	0x10	-	-	-	POS4	Yes
POSENC_HD_VELR	0x002C	0x15 0x16 0x17	-	VEL2	VEL1	VEL0	Yes
POSENC_HD__VPOSST	0x0034	0x1B 0x1C 0x1D	-	VPOS2	VPOS1	VPOS0	Yes
POSENC_HD_VPOSMT1	0x0038	0x1A 0x1B 0x1C 0x1D	VPOS3	VPOS2	VPOS1	VPOS0	Yes
POSENC_HD__VPOSMT2	0x003C	0x19	-	-	-	VPOS4	Yes
POSENC_HD_VPOSCRC	0x0040	0x1E 0x1F	-	-	VPOS_CRC_H	VPOS_CRC_L	Yes
POSENC_HD_PCBUFA	0x0044	0x20 0x21	-	-	PC_BUFFER1	PC_BUFFER0	No
POSENC_HD_PCBUFB	0x0048	0x20 0x21 0x22 0x23	PC_BUFFER3	PC_BUFFER2	PC_BUFFER1	PC_BUFFER0	No
POSENC_HD_PCBUFC	0x004C	0x24 0x25 0x26 0x27	PC_BUFFER7	PC_BUFFER6	PC_BUFFER5	PC_BUFFER4	No
POSENC_HD_LOFFR	0x0050	0x28	-	-	-	PC_ADD_H	No

Table 664. Hiperface-DSL Address conversion table (continued)

Name	32-bit address	Hiperface-DSL addresses	Byte 3	Byte 2	Byte 1	Byte 0	Freeze
POSENC_HD_PC_ADDR	0x0054	0x28 0x29	-	-	PC_ADD_H	PC_ADD_L	No
POSENC_HD_PC_OFFR	0x0058	0x2A 0x2B	-	-	PC_OFF_H	PC_OFF_L	No
POSENC_HD_PC_ADDOFFR	0x005C	0x28 0x29 0x2A 0x2B	PC_OFF_H	PC_OFF_L	PC_ADD_H	PC_ADD_L	No
POSENC_HD_PC_CTRLR	0x0060	0x2C	-	-	-	PC_CTRL	No
POSENC_HD_PIPE_SR	0x0064	0x2D	-	-	-	PIPE_SR	No
POSENC_HD_PIPE_D	0x0068	0x2E	-	-	-	PIPE_D	No
POSENC_HD_PC_DATA	0x006C	0x2F	-	-	-	PC_DATA	No
POSENC_HD_SAFE_CTRL	0x0070	0x35	-	-	-	SAFE_CTRL	No
POSENC_HD_SAFE_SUM	0x0074	0x36	-	-	-	SAFE_SUM	No
POSENC_HD_S_PC_DATA	0x0078	0x37	-	-	-	S_PC_DATA	No
POSENC_HD_ACC_EE_CNT	0x007C	0x38	-	-	-	ACC_EE_CNT	No
POSENC_HD_MAXACC	0x0080	0x39	-	-	-	MAXACC	No
POSENC_HD_MAXDEV	0x0084	0x3A 0x3B	-	-	MAXDEV_H	MAXDEV_L	Yes
POSENC_HD_EVENT_S	0x0088	0x3D	-	-	-	EVENT_S	No
POSENC_HD_MASK_S	0x008C	0x3E	-	-	-	MASK_S	No
POSENC_HD_DUMMY	0x0090	0x3F	-	-	-	DUMMY	No
POSENC_HD_ENC_ST0	0x0094	0x40	-	-	-	ENC_ST0	No
POSENC_HD_ENC_ST1	0x0098	0x41	-	-	-	ENC_ST1	No
POSENC_HD_ENC_ST2	0x009C	0x42	-	-	-	ENC_ST2	No

Table 664. Hiperface-DSL Address conversion table (continued)

Name	32-bit address	Hiperface-DSL addresses	Byte 3	Byte 2	Byte 1	Byte 0	Freeze
POSENC_HD_ENC_ST3	0x00A0	0x43	-	-	-	ENC_ST3	No
POSENC_HD_ENC_ST4	0x00A4	0x44	-	-	-	ENC_ST4	No
POSENC_HD_ENC_ST5	0x00A8	0x45	-	-	-	ENC_ST5	No
POSENC_HD_ENC_ST6	0x00AC	0x46	-	-	-	ENC_ST6	No
POSENC_HD_ENC_ST7	0x00B0	0x47	-	-	-	ENC_ST7	No
POSENC_HD_SRSSI	0x00B4	0x7C	-	-	-	SRSSI	No
POSENC_HD_MAIL	0x00B8	0x7E	-	-	-	MAIL	No
POSENC_HD_PING	0x00BC	0x7F	-	-	-	PING	No

1. Refer to *RELEASE register access* for details.

The read and write mechanisms for accessing the Hiperface-DSL registers are different and are described in the sections below.

Write accesses

The write accesses to Hiperface-DSL 8-bit registers are directly done in the 0x0000-0x00FF memory region, using addresses defined in *Table 664*. The write accesses are posted, so that the APB bus can be released immediately. It allows the CPU to resume operation without having to wait until the write is completed: a write access can last up to 8 Hiperface-DSL kernel clock cycles (107 ns) per byte.

A single write access can be posted at once: a 2nd write request is ignored if the 1st one is not completed.

It is thus mandatory to make sure the previous write command is completed before a new write request, with the following sequence:

1. Read the WRBSY bit in POSENC_SR1 to verify it is reset.
2. If the WRBSY bit is set, poll it until it is reset (or postponed the write access).
3. Write the Hiperface target register in the 0x0000-0x00FF memory range.

The steps 1 and 2 are optional and can be skipped if the write interval is guaranteed to be above the write duration, for instance with a timer-controlled periodic interrupt service routine.

It is also possible to enable an interrupt for being notified that a write transaction is completed using the Write transfer complete interrupt WRTC.

Note: A write access to Hiperface-DSL register does not prevent another access to be made to the POSENC common registers (for instance for preparing a read command).

Read accesses

The Hiperface read accesses are handled indirectly, with pre-configured read requests commands.

Similarly to write transactions, this is for the APB bus to be released immediately, without having to wait for the POSENC to complete the read transaction. A read access can last up to 3 cycles Hiperface-DSL kernel clock cycles per byte, plus a 4 clock cycle fixed overhead.

The read results are available in the POSENC memory space, from 0x3800 to 0x3FFF, in dedicated read registers.

Note: A read access to the 0x0000 to 0x00FF write region is ignored, and result is null.

The POSENC provides two options for read accesses:

- software driven read accesses: this option gives access to any Hiperface-DSL register. It requires two registers, one for specifying the address to be read and one holding the result once the read access is completed. This mechanism is non-blocking, which lets the choice for polling or interrupt-driven data read.
- Event-driven read accesses: this mechanism can further decrease the CPU burden for the frequently accessed data (fast position, speed, safe position 1 and safe position CRC). Upon arrival of a new frame, the read access is automatically done by the POSENC and an interrupt is issued once data are available in dedicated registers than are accessible without extra latency.

Software-driven read accesses

The address of the Hiperface-DSL registers to be read must be written in the POSENC_D0AR, and optionally in POSENC_D1AR registers, using address translation*Table 664*.

The read command (from POSENC to Hiperface-DSL IP core) is issued when writing the POSENC_ARx registers. When the read is completed, the result is stored in the POSENC_D0R (or POSENC_D1R) registers and D0RTCF (or D1RTCF) bit is set. An interrupt can also be issued upon completion of the read command.

To make sure the previous read or write transaction is completed before a new read request, the following sequence must be done before any read command:

1. read the D0RDBSY (or D1RDBSY) bit in POSENC_SR1 to verify it is reset
2. If the D0RDBSY (or D1RDBSY) bit is set, poll it until it is reset or postpone the read to the Hiperface-DSL (it is also possible to enable an interrupt for being notified that the read is completed using the D0RTC or D1RTC Data read transfer Complete interrupt)

Note: It is however advised to avoid situations where concurrent read and write threads are executed, and to have read and write software routines making sure that the current read or write access is completed before issuing a new one. In this case, it is not necessary to poll the BSY bit before initiating a read access, thus lowering the overall read latency.

Hereafter are described programming examples for accessing one, two, or more than two registers.

Programming guidelines for a single read access:

1. Write address in POSENC_D0AR.
2. Poll D0RDBSY until set (or use the D0RTC interrupt for a non-blocking mechanism).
3. Read result in POSENC_D0R.
4. Clear D0RTCF with CD0RTCF.

Programming guidelines for a two back-to-back read accesses:

1. Write address in POSENC_D0AR.
2. Write address in POSENC_D1AR.
3. Poll D0RDBSY until reset (or use the D0RTC interrupt for a non-blocking software).
4. Read result in POSENC_D0R.
5. Clear D0RTCF with CD0RTCF.
6. Poll D1RDBSY until reset (or use the D1RTC interrupt for a non-blocking software).
7. Read result in POSENC_D1R.
8. Clear D1RTCF with CD1RTCF.

Programming guidelines for more than two back-to-back read accesses:

1. Write address in POSENC_D0AR.
2. Write address in POSENC_D1AR.
3. Poll D0RDBSY until reset (or use the D0RTC interrupt for a non-blocking software).
4. Read result in POSENC_D0R.
5. Write address in POSENC_D0AR.
6. Clear D0RTCF with CD0RTCF.
7. Poll D1RDBSY until reset (or use the D1RTC interrupt for a non-blocking software).
8. Read result in POSENC_D1R.
9. Write address in POSENC_D1AR.
10. Clear D1RTCF with CD1RTCF.
11. Go back to step 3 for further accesses or end up with step 12 to 16.
12. Poll D0RDBSY until reset (or use the D0RTC interrupt for a non-blocking software).
13. Read result in POSENC_D0R.
14. Poll D1RDBSY until reset (or use the D1RTC interrupt for a non-blocking software).
15. Read result in POSENC_D1R.
16. Clear D0RTCF and D1RTCF with CD0RTCF and CD1RTCF.

40-bit software-driven read accesses

When using single-turn 23-bit/multi-turn 12-bit encoders, fast position and safe position 1 are 40-bit wide, and must be read with two consecutive read accesses. It is mandatory to

follow the procedure here-below for the POSENC to guarantee the coherency of the data between two consecutive accesses, using a register freeze mechanism:

- The 1st data to be read, with POSENC_D0AR and POSENC_D0R registers, must be the most significant one:
  - POSENC_HD_FPOSMT2 (fast position POS4)
  - POSENC_HD_VPOS_MT2 (safe position 1 VPOS4)
- The 2nd data to be read, with POSENC_D1AR and POSENC_D1R registers, must be the least significant one:
  - POSENC_HD_FPOSMT1 (fast position POS3..0)
  - POSENC_HD_VPOS_MT1 (safe position 1 VPOS3..0)

Event-driven read accesses

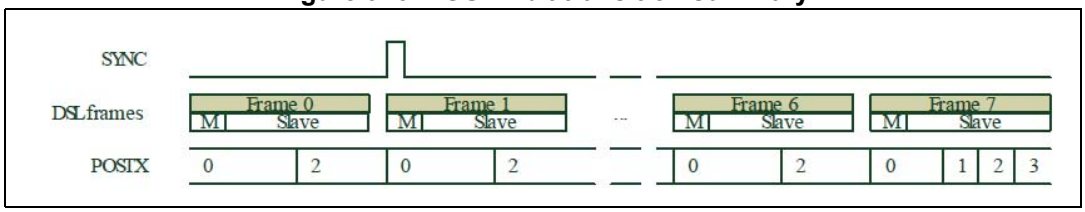
The purpose of event-driven accesses is to reduce the read latency and CPU burden, by automatically launching a read command upon reception of a frame. It allows the CPU to directly read the result in dedicated registers without having to initiate a read command to the Hiperface-DSL IP core. The following results can be read:

- Position in POSENC_HD_POS30R and POSENC_HD_POS4R registers.
- Speed in POSENC_HD_VEL20R.
- Safe position 1 in POSENC_HD_VPOS30R and POSENC_HD_VPOS4R registers.
- Safe position 1 CRC in POSENC_HD_VPOSCRCR registers.

The reading trigger event is derived from the Position transmission status bits POSTX1:POSTX0 (on the online_status_d bus). *Figure 946* summarizes the possible transitions, with the status described below:

- 00: Position request is transmitted to the DSL encoder.
- 01: Safe position 1 was received.
- 10: Fast position was received or position newly updated by estimator.
- 11: Safe position 2 was received.

Figure 946. POSTX bit transition summary



The reading event is triggered by a change of the position transmission status bits POSTX1:POSTX0, as following:

- A fast position reading (and optionally a speed reading) is triggered by a transition (00 to 10) or (01 to 10).
- A safe position 1 reading (and optionally a safe position 1 CRC reading) is triggered by a transition (00 to 01).

Note: Safe position 2 reading can only be done using the SPI access port.

The event-driven read accesses are user-configurable using the following bits in the POSENC_CR2 register:

- HDPOS[1:0] for reading the position
- HDVEL for reading speed
- HDVPOS[1:0] for safe position 1 reading
- HDVPOSCRC for safe position 1 CRC reading

An interrupt can be triggered on any of above read access completion:

- HDFPOSTC: fast position is available in POSENC_HD_POS4R and /or POSENC_HD_POS30R)
- HDVELTC: speed is available in POSENC_HD_VEL20R
- HDVPOSTC: safe position 1 is available in POSENC_HD_VPOS4R and/or POSENC_HD_VPOS30R
- HDVPOSCRCTC: safe position 1 CRC is available in POSENC_HD_VPOSCRCR.

Table 665 summarizes the 8-bit and 32-bit Hiperface-DSL register accesses in event-driven mode.

Table 665. Registers accesses in event-driven mode

Transition	Configuration	H-DSL 8-bit register accesses	H-DSL 32-bit register accesses	Interrupt	Mode
0 to 2 1 to 2	HDPOS = 1	POS0, POS1, POS2	FPOSST	FPOSRDY	Single-turn 17-bit fast position
	HDPOS = 2	POS0, POS1, POS2, POS3	FPOSMT1		Single-turn 17-bit / multi-turn 12-bit fast position
	HDPOS = 3	POS0, POS1, POS2, POS3, POS4	FPOSMT1 and FPOSMT2		Single-turn 23-bit / multi-turn 12-bit fast position
	HDVEL = 1	VEL2, VEL1, VEL0	VEL	VELRDY	Speed
0 to 1	HDVPOS = 1	VPOS0, VPOS1, VPOS2	VPOS1SST	VPOSRDY	Single-turn 17-bit safe position 1
	HDVPOS = 2	VPOS0, VPOS1, VPOS2, VPOS3	VPOS1MT1		Single-turn 17-bit / multi-turn 12-bit safe position 1
	HDVPOS = 3	VPOS0, VPOS1, VPOS2, VPOS3, VPOS4	VPOS1MT1 and VPOS1MT2		Single-turn 23-bit / multi-turn 12-bit safe position 1
	HDVPOSCRC = 1	VPOSCRC_H, VPOSCRC_L	VPOS1CRC	VPOSCRC	Safe position 1 CRC

When multiple event-driven read accesses are enabled, the accesses are done in the following sequence :

- POSENC_HD_VPOSCRCR
- POSENC_HD_VPOS4R
- POSENC_HD_VPOS30R
- POSENC_HD_VEL20R
- POSENC_HD_POS4R
- POSENC_HD_POS30R

Combining event-driven and software-driven read accesses

It is possible to combine both event-driven and software driven accesses, concurrently. A request (software or event-driven) occurring during an on-going read-access is delayed until the previous one is completed.

In case of simultaneous concurrent accesses, software accesses start first.

As software-driven (or eventually event-driven) accesses can be dynamically changed, the read commands are executed with a circular priority scheme, as following:

1. POSENC_D0R
2. POSENC_D1R
3. POSENC_HD_VPOSCRCR
4. POSENC_HD_VPOS4R
5. POSENC_HD_VPOS30R
6. POSENC_HD_VEL20R
7. POSENC_HD_POS4R
8. POSENC_HD_POS30R
9. back to step 1 (POSENC_D0R, POSENC_D1R, POSENC_HD_VPOSCRCR)

Concurrent read and write accesses

If a read access (software or event-driven) is initiated while a write access is ongoing (or simultaneously with a write access), the read access is postponed and delayed until the write is completed.

If a write access is done while a software-driven or an event-driven read access is ongoing, the write access is postponed and delayed until the circular read sequence is completed, as follows:

1. POSENC_D0R (read)
2. POSENC_D1R (read)
3. POSENC_HD_VPOSCRCR (read)
4. POSENC_HD_VPOS4R (read)
5. POSENC_HD_VPOS30R (read)
6. POSENC_HD_VEL20R (read)
7. POSENC_HD_POS4R (read)
8. POSENC_HD_POS30R (read)
9. Pending Write access, if any
10. back to step 1 (POSENC_D0R, POSENC_D1R, POSENC_HD_VPOSCRCR)

This represents the worst case situation, for what about latency: a write access requested while all read accesses are pending. For instance, if no event-driven read accesses are pending, the write access is done following the D0 (or eventually D1) software-driven read.

Data coherency in event-driven read mode

The POSENC ensures the coherency between data, by maintaining the hostd_f in between successive read accesses.

Table 666 and *Table 667* below details the list of registers accessed while the freeze signal is asserted, when this implies writes to multiple 32-bit registers. For a single 32-bit register, one can refer to *Table 664* (Freeze column).

Note: The freeze signal is always released in-between fast position / velocity readings and safe position/ safe CRC readings.

Table 666. Fast position and velocity registers freeze in event-driven mode

Mode	HDPOS	HDVEL	List of registers accessed while freeze signal is maintained asserted
Single-turn 17-bit fast position and speed	1	1	POS0, POS1, POS2, VEL2, VEL1, VEL0
Single-turn 17-bit/multi-turn 12-bit fast position and speed	2	1	POS0, POS1, POS2, POS3, VEL2, VEL1, VEL0
Single-turn 23-bit / multi-turn 12-bit fast position	3	0	POS0, POS1, POS2, POS3, POS4
Single-turn 23-bit / multi-turn 12-bit fast position and speed		1	POS0, POS1, POS2, POS3, POS4, VEL2, VEL1, VEL0

Table 667. Safe position and CRC registers freeze in event-driven mode

Mode	HDVPOS	HDVPOSCRC	List of registers accessed while freeze signal is maintained asserted
Single-turn 17-bit safe position 1 and CRC	1	1	VPOS0, VPOS1, VPOS2, VPOSCRC_H, VPOSCRC_L
Single-turn 17-bit/multi-turn 12-bit safe position 1 and CRC	2	1	VPOS0, VPOS1, VPOS2, VPOS3, VPOSCRC_H, VPOSCRC_L
Single-turn 23-bit/multi-turn 12-bit safe position 1	3	0	VPOS0, VPOS1, VPOS2, VPOS3, VPOS4
Safe position 1 CRC		1	VPOS0, VPOS1, VPOS2, VPOS3, VPOS4 VPOSCRC_H, VPOSCRC_L

Software freeze command

The HDFRZ control bit in POSENC_CR2 registers provides a mean to force the Hiperface-DSL freeze signal, and override the freeze signal coming from the event-driven interface.

This bit can be enabled when it is necessary to freeze Hiperface-DSL during multiple software-driven read accesses.

Note: The event-driven read command are served while the HDFRZ bit is set. The registers content will correspond to the point at which the HDFRZ bit was set.

RELEASE register access

The RELEASE register is a 3-byte deep FIFO, which includes day, month, and year. To get the full date, it is mandatory to have the freeze signal asserted and released between the accesses to the RELEASE register to change the FIFO pointer.

It is also necessary to first read the VERSION register to enable the access to the release register.

This procedure is supported by the POSENC. A read access to the 32-bit register POSENC_HD_VRS_RELR provides the following data at once:

- VERSION
- RELEASE (year)
- RELEASE (month)
- RELEASE (day)

SPI-based read accesses

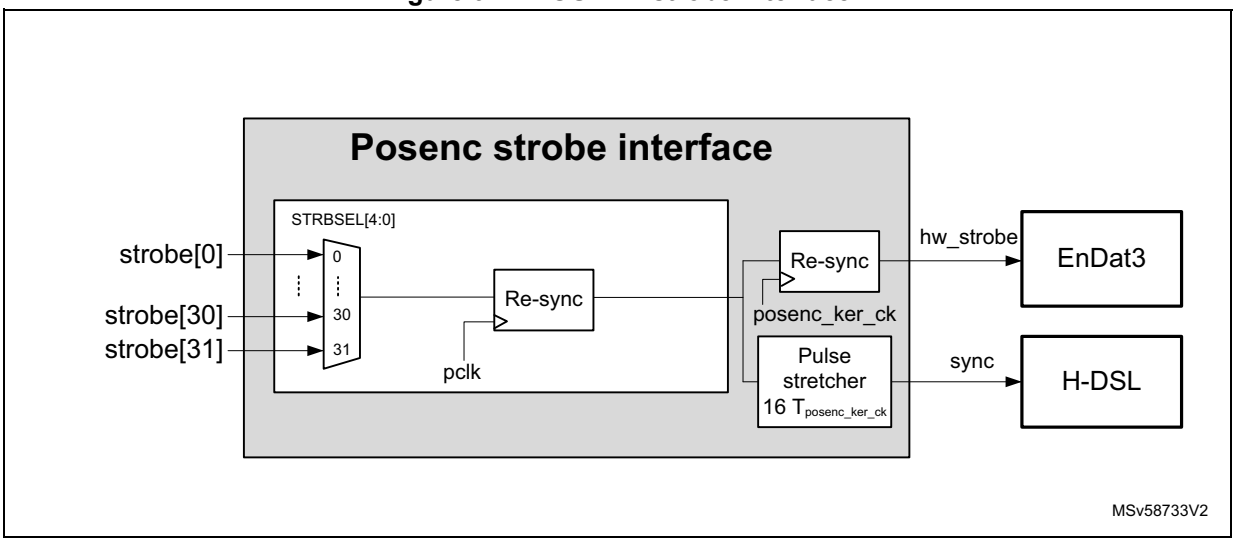
Access to the Hiperface-DSL Safe 2 interface register are performed using the SPI port, using the protocol defined in Hiperface-DSL technical specification.

57.3.7 POSENC strobe interface

The position reading can be initiated by hardware, to synchronize it with a control cycle or with a timer in charge of the PWM generation and motor drive.

This is achieved with the strobe interface described in *Figure 947*.

Figure 947. POSENC strobe interface



The interface features a 32:1 selection multiplexer controlled with the STRBSEL[4:0] bitfield in the POSENC_CR register.

The interface is also responsible for reshaping and resynchronizing the trigger signals to align them with the selected IP core requirements.

Table 668 below provides the minimum pulse width requirements on the strobe[31:0] interface.

Table 668. Strobe interface pulse-width requirements

Strobe Source	Conditions (frequency)	Min pulse width
Off-chip (using EXTI) PLAy (asynchronous mode)	-	2x T_{posenc_pclk}
On-chip peripherals (including PLAy synchronous mode)	Fposenc_pclk >= Fsource	1x T_{posenc_pclk}

For instance, if both the POSENC APB clokc and TIMx peripheral kernel clock are 100 MHz APB, the minimum pulse width to be generated on the TIMx trgo output is 10 ns (1x T_{posenc_pclk} = 1 x$T_{tim_ker_ck}$).

On the other hand, if the POSENC APB clock is 100 MHz and the timer kernel clock is 250 MHz, the minimum pulse width to be generated on the TIMx trgo output will be 10 ns (1x T_{posenc_pclk}), correspond to 2.5 $T_{tim_ker_ck}$. The timer must thus be programmed to issue a pulse width of 2.5 $T_{tim_ker_ck}$.

Caution: In this case the following TIMx events (generating a single clock period pulse) cannot be used for master mode selection: “Reset”, “Enable”, “Update”, “Compare pulse”. It is mandatory to use “Compare” mode to generate programmable width signals.

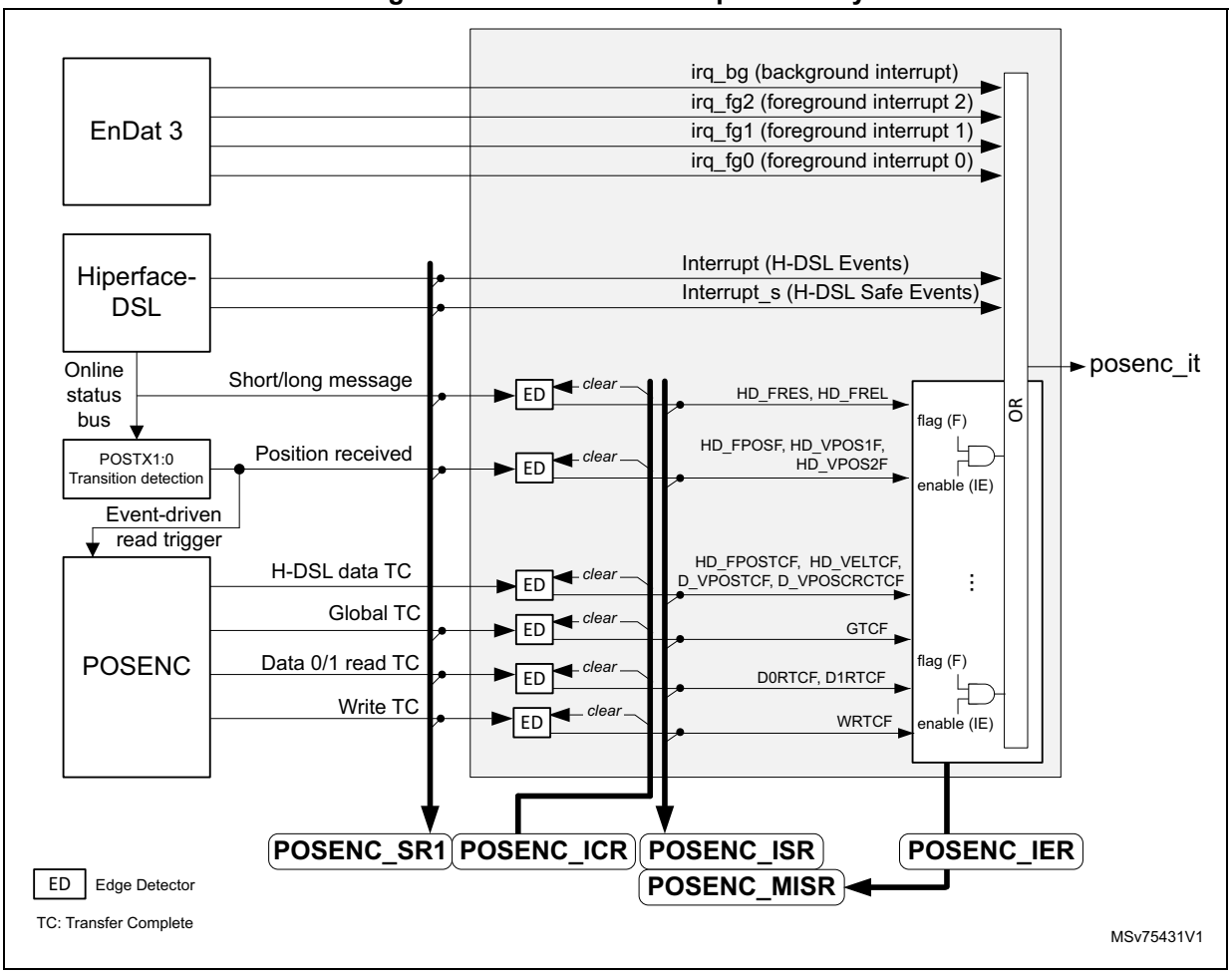
Caution: The HRTIM ADC trigger length is 1x $T_{hrtim_ker_ck}$ and cannot be extended.

57.4 POSENC interrupts

The POSENC provides an interrupt line, which gathers the interrupts from the various IP cores and from the common interface. All interrupt sources are ORed, and can be enabled individually, either in the IP cores or with the POSENC_IER register.

Figure 948 provides a summary of the interrupt sources and control and status logic.

Figure 948. POSENC interrupt summary



EnDat3 interrupts

The four EnDat3 interrupts (irq_bg and irq_fg2..0) are ORed and propagated on the posenc_it line.

Hiperface-DSL IP core interrupts

The Hiperface-DSL interrupt and interrupt_s signals are ORed and propagated on the posenc_it line.

Hiperface-DSL Short message interrupt (HDFRES)

This interrupt is generated using the online_status_1[0] bit (FRES, Channel status for the “short message”), when detecting a 0 to 1 transition. It indicates that the channel changed from ”in use” to “free”, and that the short message has been received. It is then up to software to read the requested data using a software-driven read procedure.

Hiperface-DSL Long message interrupt (HDFREL)

This interrupt is generated using the online_status_d[1] bit (FREL, Channel status for the “long message”), when detecting a 0 to 1 transition. It indicates that the channel changed

from ”in use” to “free”, and that the long message has been received. It is then up to software to read the requested data using a software-driven read procedure.

Hiperface-DSL fast position interrupt (HDFPOS)

This interrupt is generated using the online_status_d[15:0] bus, when detecting a bit change on Position transmission status (bits 7 and 6, POSTX).

It is triggered by a transition from:

- 00 (Position request transmitted to the DSL encoder) or 01 (safe position 1 received)
- to 10: Fast position was received or position newly updated by estimator

The purpose of this interrupt is to signal to the CPU that a frame containing the position was received. It is then up to software to read the requested data using the software-driven read procedure.

Hiperface-DSL safe position 1 interrupt (HDVPOS1)

This interrupt is generated using the online_status_d[15:0] bus, when detecting a bit change on Position transmission status (bits 7 and 6, POSTX).

This corresponds to a transition from:

- 00 (Position request transmitted to the DSL encoder)
- to 01: Safe position 1 was received

The purpose of this interrupt is to signal to the CPU that a frame containing the safe position 1 was received. It is then up to software to read the requested data using the software-driven read procedure.

Hiperface-DSL safe position 2 interrupt (HDVPOS2)

This interrupt is generated using the online_status_d[15:0] bus, when detecting a bit change on Position transmission status (bits 7 and 6, POSTX).

This corresponds to a transition from:

- 10: Fast position was received or position newly updated by estimator, or 00 (Position request transmitted to the DSL encoder)
- to 11: Safe position 2 was received

The purpose of this interrupt is to signal to the CPU that a frame containing the safe position 2 was received.

Note: Safe position 2 reading can only be done by an external device (such as Safe CU) using the SPI access port.

Hiperface-DSL Data ready interrupts

Four interrupts can be issued for event-driven read accesses, to indicate that the new data was read and made available in the POSENC Hiperface-DSL dedicated registers, as follows:

- HDFPOSTC: fast position is available in POSENC_HD_POS4 and /or POSENC_HD_POS3_0).
- HDVELTC: speed is available in POSENC_HD_VEL2_0.
- HDVPOSTC: safe position 1 is available in POSENC_HD_VPOS4 and/or POSENC_HD_VPOS3_0.
- HDVPOSCRCTC: safe position 1 CRC is available in POSENC_HD_VPOSCRC.

In the corresponding IRQ handler, the CPU only needs to read the listed registers without any software read-access to the Hiperface-DSL master IP core internal registers.

These interrupts must be distinguished from the HD_FPOS, HD_VPOS1 and HD_VPOS2, only indicating that a new frame arrived.

Data read transfer complete interrupts (DORTC, D1RTC)

Two interrupts can be issued on, software-driven read accesses, to indicate that the new data was read and made available in the POSENC_D0R and POSENC_D1R registers.

Global transfer Complete interrupt

This interrupt indicates that:

- a write access is completed
- a read access is completed

Note: When multiple back-to-back read accesses are issued, the interrupt will only be asserted at the end of the very last access. It is thus necessary in this case to use the D0RTC or D1RTC interrupts to have an interrupt issued on each and every new reading.

Table 669 provides a summary of the POSENC interrupt sources.

Table 669. Interrupt summary

Interrupt event	Status flag	Enable control bit	Interrupt clear method	Purpose	EnDAT3	Hiperface-DSL
irq_bg irq_fg0 irq_fg1 irq_fg2	-	-	-	EnDat3 IP-core interrupts	X	-
interrupt	-	-	-	Hiperface-DSL IP-core interrupts	-	X
interrupt_s	INT INT_S	-	-	For polling purposes, to discriminate IP-core and POSENC generated Hiperface-DSL interrupts	-	X
HDFRES	HDFRESF HDFRESMF	HDFRESIE	Write 1 in HDCFRESF	Indicates short message received	-	X
HDFREL	HDFRELF HDFRELMF	HDFRELIE	Write 1 in HDCFRELF	Indicates long message received	-	X
HDVPOS2	HDVPOS2F HDVPOS2MF	HDVPOS2IE	Write 1 in HDCVPOS2F	Indicates safe position 2 received	-	X

Table 669. Interrupt summary (continued)

Interrupt event	Status flag	Enable control bit	Interrupt clear method	Purpose	EnDAT3	Hiperface-DSL
HDVPOS1	HD_VPOS1F HD_VPOS1MF	HDVPOS1IE	Write 1 in HDCVPOS1F	Indicates safe position 1 received	-	X
HDFPOS	HD_FPOSF HD_FPOSMF	HDFPOSIE	Write 1 in HDCFPOSF	Indicates fast position received	-	X
HDFPOSTC	HDFPOSTCF HD_FPOSTCMF	HDFPOSTCIE	Write 1 in HDCFPOSTCF	Indicates fast position received and stored in POSENC_HD_POS30R / POSENC_HD_POS4R	-	X
HDVELTC	HD_VELTCF HD_VELTCMF	HD_VELTCIE	Write 1 in HD_CVELTCF	Indicates Speed received and stored in POSENC_HD_VEL20R	-	X
HDVPOSTC	HDVPOSTCF HDVPOSTCMF	HDVPOSTCIE	Write 1 in HDCVPOSTCF	Indicates safe position 1 received and stored in POSENC_HD_VPOS30R / POSENC_HD_VPOS4R	-	X
HDVPOSCRCTC	HDVPOSCRCTCF HDVPOSCRCTCMF	HDVPOSCRCTCIE	Write 1 in HDCVPOSCRCTCF	Indicates safe position 1 CRC received and stored in POSENC_HD_VPOSCRCR	-	X
D1RTC	D1RTCF D1RTCMF	D1RTCIE	Write 1 in CD1RDYF	-	-	X
	D1RDBSY	-	-	For polling purpose.	-	X
D0RTC	D0RTCF D0RTCMF	D0RTCIE	Write 1 in CD0RDYF	-	-	X
	D0RDBSY	-	-	For polling purpose	-	X

Table 669. Interrupt summary (continued)

Interrupt event	Status flag	Enable control bit	Interrupt clear method	Purpose	EnDAT3	Hiperface-DSL
GTC	GTCF GTCMF	GTCIE	CGTCF	Global transfer completion (no pending read or write access)	-	X
	GBSY	-	-	Indicates a read or write access is on-going (for polling purposes)	-	X

57.5 POSENC initialization

It is mandatory to program the protocol at first, using the ENCSEL[3:0] bitfield in POSENC_CR1, before doing any access to other POSENC registers.

The POSENC_CR1 is locked and read-only once a value different than 0xF is programmed in ENCSEL[3:0].

Consequently, the very first initialization step can be done in two ways:

- A single write accesses to POSENC_CR1, where all bitfields are defined at once (STRBSEL[3:0] and ENCSEL[3:0]
- One or to write accesses to POSENC_CR1 to define STRBSEL[3:0] if needed, followed by a last write access to ENCSEL[3:0]

From this point onward the other POSENC registers can be programmed, in any order.

57.6 POSENC registers

57.6.1 POSENC control register 1 (POSENC_CR1)

Address offset: 0x3800

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STRBSEL[4:0]					Res.	Res.	Res.	NSTRB RSYNC	ENCSEL[3:0]			
			rw	rw	rw	rw	rw				rw	rwo	rwo	rwo	rwo

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **STRBSEL[4:0]**:Strobe selection
Selects strobe[31:0] input
00000: strobe0
00001: strobe1
...
11111: strobe31
Refer to *Table 668: POSENC strobe sources* for interconnects list.

Note: This bitfield cannot be modified once the protocol has been selected (ENCSEL[3:0] is different from 0xF).

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **NSTRBRSYNC**: Strobe re-synchronization
0: Strobe re-synchronization enabled
1: Reserved, must be kept at reset value.

Bits 3:0 **ENCSEL[3:0]**: Encoder selection
This bitfield selects the protocol:
0000: EnDat3
0010: Not used
0100: Hiperface-DSL
1111: No protocol selected
Others: reserved

Note: This bit cannot be modified once the protocol has been selected (ENCSEL[3:0] is different from 0xF)

57.6.2 POSENC control register 2 (POSENC_CR2)

Address offset: 0x3804

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDFRZ
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HDVPOSCRC	HDVPOS[1:0]		HDVEL	HDPOS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw								

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **HDFRZ**: Hiperface-DSL freeze
0: Hiperface-DSL registers are not freezed.
1: Hiperface-DSL registers are freezed.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **HDVPOSCRC**: Hiperface-DSL velocity event-driven accesses
0: No safe position 1 CRC reading
1: Safe position 1 CRC reading (VPOSCRC stored in POSENC_HD_VPOSCRC)

Bits 12:11 **HDVPOS[1:0]**: Hiperface-DSL fast position event-driven accesses
00: No safe position 1 reading
01: Single-turn 17-bit safe position 1 reading (VPOS0..2 stored in POSENC_HD_VPOS3_0)
10: Single-turn 17-bit/multi-turn 12-bit safe position 1 reading (VPOS0..3 stored in POSENC_HD_VPOS3_0)
11: Single-turn 23-bit/multi-turn 12-bit safe position 1 reading (VPOS0..4 stored in POSENC_HD_VPOS3_0 and POSENC_HD_VPOS4)

Note: This bitfield must not be modified during run-time, once Hiperface-DSL reception has started.

Bit 10 **HDVEL**: Hiperface-DSL Velocity event-driven accesses
0: No velocity reading
1: Velocity reading (VEL0..2 stored in POSENC_HD_VEL2_0)

Bits 9:8 **HDPOS[1:0]**: Hiperface-DSL fast position event-driven accesses
00: No position reading
01: Single-turn 17-bit position reading (POS0..2 stored in POSENC_HD_POS3_0)
10: Single-turn 17-bit/multi-turn 12-bit position reading (POS0..3 stored in POSENC_HD_POS3_0)
11: Single-turn 23-bit/multi-turn 12-bit position reading (POS0..4 stored in POSENC_HD_POS3_0 and POSENC_HD_POS4)
Note: This bitfield must not be modified during run-time, once Hiperface-DSL reception has started.

Bits 7:0 Reserved, must be kept at reset value.

57.6.3 POSENC interrupt enable register (POSENC_IER)

Address offset: 0x3810

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDFRESIE	HDFRELIE	HDVPOS2IE	HDVPOS1IE	HDFPOSIE
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HDFPOSTCIE	Res.	HDVELTCIE	HDVPOSTCIE	Res.	HDVPOSCRCTCIE	D1RTCIE	D0RTCIE	Res.	Res.	Res.	Res.	Res.	Res.	WRTCIE	GTCIE
rw		rw	rw		rw	rw	rw							rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HDFRESIE**: Hiperface-DSL short message interrupt enable
0: Hiperface-DSL short message interrupt disabled
1: Hiperface-DSL short message interrupt enabled

Bit 19 **HDFRELIE**: Hiperface-DSL long message interrupt enable
0: Hiperface-DSL long message interrupt disabled
1: Hiperface-DSL long message interrupt enabled

Bit 18 **HDVPOS2IE**: Hiperface-DSL safe position 2 interrupt enable
0: Hiperface-DSL safe position 2 interrupt disabled
1: Hiperface-DSL safe position 2 interrupt enabled

Bit 17 **HDVPOS1IE**: Hiperface-DSL safe position 1 interrupt enable
0: Hiperface-DSL safe position 1 interrupt disabled
1: Hiperface-DSL safe position 1 interrupt enabled

Bit 16 **HDFPOSIE**: Hiperface-DSL fast position interrupt enable
0: Hiperface-DSL fast position interrupt disabled
1: Hiperface-DSL fast position interrupt enabled

Bit 15 **HDFPOSTCIE**: Hiperface-DSL fast position transfer complete interrupt enable
0: Hiperface-DSL fast position transfer complete interrupt disabled
1: Hiperface-DSL fast position transfer complete interrupt enabled

Bit 14 Reserved, must be kept at reset value.

Bit 13 **HDVELTCIE**: Hiperface-DSL velocity transfer complete interrupt enable
0: Hiperface-DSL velocity transfer complete interrupt disabled
1: Hiperface-DSL velocity transfer complete interrupt enabled

Bit 12 **HDVPOSTCIE**: Hiperface-DSL safe position 1 transfer complete interrupt enable
0: Hiperface-DSL safe position 1 transfer complete interrupt disabled
1: Hiperface-DSL safe position 1 transfer complete interrupt enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **HDVPOSCRCTCIE**: Hiperface-DSL safe position 1 CRC transfer complete interrupt enable
0: Hiperface-DSL safe position 1 CRC transfer complete interrupt disabled
1: Hiperface-DSL safe position 1 CRC transfer complete interrupt enabled

Bit 9 **D1RTCIE**: Data 1 Read transfer complete interrupt enable
0: Data 1 transfer complete interrupt disabled
1: Data 1 transfer complete interrupt enabled

Bit 8 **D0RTCIE**: Read Data 0 transfer complete interrupt enable
0: Data 0 transfer complete interrupt disabled
1: Data 0 transfer complete interrupt enabled

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **WRTCIE**: Write transfer complete interrupt enable
0: Write transfer complete interrupt disabled
1: Write transfer complete interrupt enabled

Bit 0 **GTCIE**: Global Transfer complete interrupt enable
0: Global transfer complete interrupt disabled
1: Global transfer complete interrupt enabled

57.6.4 POSENC interrupt clear register (POSENC_ICR)

Address offset: 0x3814

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDCFRESF	HDCFRELF	HDCVPOS2F	HDCVPOS1F	HDCFPOSF
											w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HDCFPOSTCF	Res.	HDCVELTCF	HDCVPOSTCF	Res.	HDCVPOSCRCTCF	CD1RTCF	CD0RTCF	Res.	Res.	Res.	Res.	Res.	Res.	CWRTCF	CGTCF
w		w	w		w	w	w							w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HDCFRESF**: Clear Hiperface-DSL short message interrupt flag
Writing 1 in this bit clears the HDFRESF flag.

Bit 19 **HDCFRELF**: Clear Hiperface-DSL long message interrupt flag
Writing 1 in this bit clears the HDFRELF flag.

Bit 18 **HDCVPOS2F**: Clear Hiperface-DSL safe position 2 interrupt flag
Writing 1 in this bit clears the HDVPOS2F flag.

Bit 17 **HDCVPOS1F**: Clear Hiperface-DSL safe position 1 interrupt flag
Writing 1 in this bit clears the HDVPOS1F flag.

Bit 16 **HDCFPOSF**: Clear Hiperface-DSL fast position interrupt flag
Writing 1 in this bit clears the HDFPOSF flag.

Bit 15 **HDCFPOSTCF**: Clear Hiperface-DSL fast position transfer complete interrupt flag
Writing 1 in this bit clears the HDFPOSTCF flag.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **HDCVELTCF**: Clear Hiperface-DSL velocity transfer complete interrupt flag
Writing 1 in this bit clears the HDVELTCF flag.

Bit 12 **HDCVPOSTCF**: Clear Hiperface-DSL safe position 1 transfer complete interrupt flag
Writing 1 in this bit clears the HDVPOSTCF flag.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **HDCVPOSCRCTCF**: Clear Hiperface-DSL safe position 1 CRC transfer complete interrupt flag
Writing 1 in this bit clears the VPOSCRCTCF flag.

Bit 9 **CD1RTCF**: Clear data 1 read transfer complete interrupt flag
Writing 1 in this bit clears the D1RTCF flag.

Bit 8 **CD0RTCF**: Clear data 0 read transfer complete interrupt flag
Writing 1 in this bit clears the D0RTCF flag.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **CWRTCF**: Clear write transfer complete interrupt flag
Writing 1 in this bit clears the WRTCF flag.

Bit 0 **CGTCF**: Clear transfer complete interrupt flag
Writing 1 in this bit clears the GTCF flag.

57.6.5 POSENC masked interrupt status register (POSENC_MISR)

Address offset: 0x3818

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDFRESMF	HDFRELMF	HDVPOS2MF	HDVPOS1MF	HDFPOSMF
											w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HDFPOSRTCMF	Res.	HDVELRTCMF	HDVPOSRTCMF	Res.	HDVPOSCRCTCMF	D1RTCMF	D0RTCMF	Res.	Res.	Res.	Res.	Res.	Res.	WRTCMF	GTCMF
r		r	r		r	r	r							r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HDFRESMF**: Hiperface-DSL short message interrupt masked flag
This flag is set by hardware when the Hiperface-DSL short message is received, and HDFRESIE bit is set.

Bit 19 **HDFRELMF**: Hiperface-DSL long message interrupt masked flag
This flag is set by hardware when the Hiperface-DSL long message is received, and HDFRELIE bit is set.

Bit 18 **HDVPOS2MF**: Hiperface-DSL safe position 2 interrupt masked flag
This flag is set by hardware when the Hiperface-DSL safe position 2 is received, and HDVPOS2IE bit is set.

Bit 17 **HDVPOS1MF**: Hiperface-DSL safe position 1 interrupt masked flag
This flag is set by hardware when the Hiperface-DSL safe position 1 is received, and HDVPOS1IE bit is set.

Bit 16 **HDFPOSMF**: Hiperface-DSL fast position interrupt masked flag
This flag is set by hardware when the Hiperface-DSL fast position is received, and HDFPOSIE bit is set.

Bit 15 **HDFPOSRTCMF**: Hiperface-DSL fast position 1 ready interrupt masked flag
This flag is set by hardware when fast position has been read and stored in POSENC_HD_POS3_0, and HDFPOSRTCIE bit is set.
When HDPOS[1:0] = 11, POSENC_HD_POS4 is also updated.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **HDVELRTCMF**: Hiperface-DSL velocity ready interrupt masked flag
This flag is set by hardware when velocity has been read and stored in POSENC_HD_VEL2_0, and HDVELRTCIE bit is set.

Bit 12 **HDVPOSRTCMF**: Hiperface-DSL safe position 1 transfer complete interrupt masked flag
This flag is set by hardware when safe position 1 has been read and stored in POSENC_HD_VPOS3_0, and HDVPOSRTCIE bit is set.
When HDVPOS[1:0] = 11, POSENC_HD_VPOS4 is also updated.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **HDVPOSCRCTCMF**: Hiperface-DSL safe position 1 CRC transfer complete interrupt masked flag
This flag is set by hardware when safe position 1 CRC has been read and stored in POSENC_HD_VPOSCRC, and HDVPOSCRCTCIE bit is set.

Bit 9 **D1RTCMF**: Data 1 read transfer complete masked flag
This flag is set by hardware when the data 1 read transaction is completed (data is available in POSENC_D1R register) and D1RTCIE bit is set.

Bit 8 **D0RTCMF**: Data 0 read transfer complete masked flag
This flag is set by hardware when the data 0 read transaction is completed (data is available in POSENC_D0R register) and D0RTCIE bit is set.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **WRTCMF**: Write transfer complete masked flag
This flag is set by hardware when the write transaction is completed and WRTCIE bit is set.

Bit 0 **GTCMF**: Global transfer complete masked flag
This flag is set by hardware when a Read or Write access to Hiperface-DSL registers is completed and GTCIE bit is set. When multiple back-to back read accesses are done, this flag is set on completion of the last read access.

57.6.6 POSENC interrupt status register (POSENC_ISR)

Address offset: 0x381C

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDFRESF	HDFRELF	HDVPOS2F	HDVPOS1F	HDFPOSF
											w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HDFPOSRTCF	Res.	HDVELRTCF	HDVPOSRTCF	Res.	HDVPOSCRCTCF	D1RTCF	D0RTCF	Res.	Res.	Res.	Res.	Res.	Res.	WRTCF	GTCF
r		r	r		r	r	r							r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HDFRESF**: Hiperface-DSL short message interrupt flag
This flag is set by hardware when the Hiperface-DSL short message is received.

Bit 19 **HDFRELF**: Hiperface-DSL long message interrupt masked flag
This flag is set by hardware when the Hiperface-DSL long message is received.

Bit 18 **HDVPOS2F**: Hiperface-DSL safe position 2 interrupt flag
This flag is set by hardware when the Hiperface-DSL safe position 2 is received.

Bit 17 **HDVPOS1F**: Hiperface-DSL safe position 1 interrupt flag
This flag is set by hardware when the Hiperface-DSL safe position 1 is received.

Bit 16 **HDFPOSF**: Hiperface-DSL fast position interrupt flag
This flag is set by hardware when the Hiperface-DSL fast position is received.

Bit 15 **HDFPOSRTCF**: Hiperface-DSL fast position transfer complete interrupt flag
This flag is set by hardware when fast position has been read and stored in POSENC_HD_POS3_0.
When HDPOS[1:0] = 11, POSENC_HD_POS4 is also updated.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **HDVELRTCF**: Hiperface-DSL velocity transfer complete interrupt flag
This flag is set by hardware when velocity has been read and stored in POSENC_HD_VEL2_0

Bit 12 **HDVPOSRTCF**: Hiperface-DSL safe position 1 transfer complete interrupt flag
This flag is set by hardware when safe position 1 has been read and stored in POSENC_HD_VPOS3_0.
When HDVPOS[1:0] = 11, POSENC_HD_VPOS4 is also updated.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **HDVPOSCRCTCF**: Hiperface-DSL safe position 1 CRC transfer complete interrupt flag
This flag is set by hardware when safe position 1 CRC has been read and stored in POSENC_HD_VPOSCRC.

Bit 9 **D1RTCF**: Data 1 read transfer complete flag
This flag is set by hardware when the data 1 read transaction is completed and data is available in POSENC_D1R register.

Bit 8 **D0RTCF**: Data 0 read transfer complete flag
This flag is set by hardware when the data 0 read transaction is completed and data is available in POSENC_D0R register.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **WRTCF**: Write transfer complete flag
This flag is set by hardware when the write transaction is completed.

Bit 0 **GTCF**: Global transfer complete flag
This flag is set by hardware when a Read or Write access to Hiperface-DSL registers is completed. When multiple back-to back read accesses are done, this flag is set on completion of the last read access.

57.6.7 POSENC interrupt status register 1 (POSENC_SR1)

Address offset: 0x3820

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDINTS	HDINT	Res.	Res.	Res.	Res.
										r	r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	D1RDBSY	D0RDBSY	Res.	Res.	Res.	Res.	Res.	Res.	WRBSY	GBSY
						r	r							r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **HDINTS**: Hiperface-DSL interrupt_s line status
This status flag indicates the Hiperface-DSL safe interrupt_s line (INT_S output) status

Bit 20 **HDINT**: Hiperface-DSL interrupt line status
This status flag indicates the Hiperface-DSL interrupt line (INT output) status

Bits 19:10 Reserved, must be kept at reset value.

Bit 9 **D1RDBSY**: Data 1 read busy
This status flag is set when the data 1 read transaction is ongoing. It is reset when it is completed.

Bit 8 **D0RDBSY**: Data 0 read busy
This status flag is set when the data 0 read transaction is ongoing. It is reset when it is completed.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **WRBSY**: Write busy
This status flag is set when a write transaction is on-going. It is reset when it is completed.

Bit 0 **GBSY**: Global busy
This status flag is set when a read transaction (software or event-driven) or a write transaction is on-going.

57.6.8 POSENC interrupt status register 2 (POSENC_SR2)

Address offset: 0x3824

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ONLINESTATUS1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ONLINESTATUSD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **ONLINESTATUS1[15:0]**: Online status safe 1 interface
This bitfield is a copy of Hiperface-DSL online_status_1[15:0] bus.

Bits 15:0 **ONLINESTATUSD[15:0]**: Online status safe 1 interface
This bitfield is a copy of Hiperface-DSL online_status_d[15:0] bus.

57.6.9 POSENC interrupt status register 3 (POSENC_SR3)

Address offset: 0x3828

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ONLINESTATUS2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ONLINESTATUS2[15:0]**: Online status safe 2 interface
This bitfield is a copy of Hiperface-DSL online_status_2[15:0] bus.

57.6.10 POSENC data 0 address register (POSENC_D0AR)

Address offset: 0x3A04

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D0ADD[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **D0ADD[7:0]**: Data 0 address

This register specifies the address of the register to be read.The read access is initiated when the address is written in this register.

57.6.11 POSENC data 1 address register (POSENC_D1AR)

Address offset: 0x3A08

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D1ADD[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **D1ADD[7:0]**: Data 1 address

This register specifies the address of the register to be read. The read access is initiated when the address is written in this register.

57.6.12 POSENC data 0 register (POSENC_D0R)

Address offset: 0x3B04

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **D0[31:0]**: Data 0 value

This register holds the data read at the address specified in the POSENC_AR0 register.

57.6.13 POSENC data 1 register (POSENC_D1R)

Address offset: 0x3B08

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **D1[31:0]**: Data 1 value

This register holds the data read at the address specified in the POSENC_AR1 register.

57.6.14 POSENC Hiperface-DSL safe position 1 CRC register (POSENC_HD_VPOSCRCR)

Address offset: 0x3B0C

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VPOSCRCH[7:0]								VPOSCRCL[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **VPOSCRCH[7:0]**: Hiperface-DSL safe position 1 CRC value
This register holds the data read in VPOSCRC_H register.

Bits 7:0 **VPOSCRCL[7:0]**: Hiperface-DSL safe position 1 CRC value
This register holds the data read in VPOSCRC_L register.

57.6.15 POSENC Hiperface-DSL safe position 1 VPOS4 register (POSENC_HD_VPOS4R)

Address offset: 0x3B10

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VPOS4[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **VPOS4[7:0]**: Hiperface-DSL safe position 1 VPOS4 value
This register holds the data read in VPOS4 registers.

57.6.16 POSENC Hiperface-DSL safe position 1 VPOS3_0 register (POSENC_HD_VPOS30R)

Address offset: 0x3B14

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VPOS3[7:0]								VPOS2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VPOS1[7:0]								VPOS0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **VPOS3[7:0]**: Hiperface-DSL safe position 1 VPOS3 value
This register holds the data read in VPOS3 register.

Bits 23:16 **VPOS2[7:0]**: Hiperface-DSL safe position 1 VPOS2 value
This register holds the data read in VPOS2 register.

Bits 15:8 **VPOS1[7:0]**: Hiperface-DSL safe position 1 VPOS1 value
This register holds the data read in VPOS1 register.

Bits 7:0 **VPOS0[7:0]**: Hiperface-DSL safe position 1 VPOS0 value
This register holds the data read in VPOS0 register.

57.6.17 POSENC Hiperface-DSL safe position 1 VEL2_0 register (POSENC_HD_VEL20R)

Address offset: 0x3B18

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VEL2[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VEL1[7:0]								VEL0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **VEL2[7:0]**: Hiperface-DSL velocity VEL2 value
This register holds the data read in VEL2 register.

Bits 15:8 **VEL1[7:0]**: Hiperface-DSL velocity VEL1 value
This register holds the data read in VEL1 register.

Bits 7:0 **VEL0[7:0]**: Hiperface-DSL velocity VEL0 value
This register holds the data read in VEL0 register.

57.6.18 POSENC Hiperface-DSL fast position POS4 register (POSENC_HD_POS4R)

Address offset: 0x3B1C

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POS4[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **POS4[7:0]**: Hiperface-DSL fast position POS4 value
This register holds the data read in POS4 registers.

57.6.19 POSENC Hiperface-DSL fast position POS3_0 register (POSENC_HD_POS30R)

Address offset: 0x3B20

Reset value: 0x0000 0000

This register is significant only when using Hiperface-DSL (ENCSEL[3:0] = 0100)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POS3[7:0]								POS2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POS1[7:0]								POS0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **POS3[7:0]**: Hiperface-DSL fast position POS3 value
This register holds the data read in POS3 register.

Bits 23:16 **POS2[7:0]**: Hiperface-DSL fast position POS2 value
This register holds the data read in POS2 register.

Bits 15:8 **POS1[7:0]**: Hiperface-DSL fast position POS1 value
This register holds the data read in POS1 register.

Bits 7:0 **POS0[7:0]**: Hiperface-DSL fast position POS0 value
This register holds the data read in POS0 register.

57.6.20 POSENC register map

Table 670. POSENC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3800	POSENC_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCPRTCKCR[3:0]				Res.	Res.	Res.	STRBSEL[4:0]					Res.	Res.	Res.	NSTRBRSYNC	ENCSEL[3:0]			
	Reset value													0	0	0	0				0	0	0	0	0				0	1	1	1	1
0x3804	POSENC_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDFRZ	Res.	Res.	HDVPOSCRC	HDVPOS[1:0]		HDVEL	HDPOS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0			0	0	0	0	0	0								
0x3808 - 0x380C	Reserved	Res.																															
0x3810	POSENC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDFRESIE	HDFREL1IE	HDVPOS2IE	HDVPOS1IE	HDFPOSIE	HDFPOSRTCIE	Res.	HDVELRTCCIE	HDVPOSRTCIE	Res.	HDVPOSCRTCIE	D1RTCIE	D0RTCIE	Res.	Res.	Res.	Res.	Res.	Res.	WRTCIE	GTCIE
	Reset value												0	0	0	0	0	0		0	0		0	0	0							0	0
0x3814	POSENC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDCFRESF	HDCFREL1F	HDCVPOS2F	HDCVPOS1F	HDCFPOSF	HDCFPOSRTCF	Res.	HDCVELRTCF	HDCVPOSRTCF	Res.	HDCVPOSCRCTCF	CD1RTCF	CD0RTCF	Res.	Res.	Res.	Res.	Res.	Res.	CWRTCF	CGTCF
	Reset value												0	0	0	0	0	0		0	0		0	0	0							0	0
0x3818	POSENC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDFRESMF	HDFREL1MF	HDVPOS2MF	HDVPOS1MF	HDFPOSMF	HDFPOSRTCMF	Res.	HDVELTCMF	HDVPOSTCMF	Res.	HDVPOSCRCTCMF	D1RTCMF	D0RTCMF	Res.	Res.	Res.	Res.	Res.	Res.	WRTCMF	GTCMF
	Reset value												0	0	0	0	0	0		0	0		0	0	0							0	0
0x381C	POSENC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDFRESF	HDFREL1F	HDVPOS2F	HDVPOS1F	HDFPOSF	HDFPOSRTCF	Res.	HDVELTCF	HDVPOSTCF	Res.	HDVPOSCRCTCF	D1RTCF	D0RTCF	Res.	Res.	Res.	Res.	Res.	Res.	WRTCF	GTCF
	Reset value												0	0	0	0	0	0		0	0		0	0	0							0	0
0x3820	POSENC_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDINTS	HDINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D1RDBS	D0RDBS	Res.	Res.	Res.	Res.	Res.	Res.	WRBSY	GBSY
	Reset value											0	0											0	0							0	0
0x3824	POSENC_SR2	ONLINESTATUS1[15:0]																ONLINESTATUSD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3828	POSENC_SR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ONLINESTATUS2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x382C - 0x3A00	Reserved	Res.																															

Table 670. POSENC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3A04	POSENC_D0AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D0ADD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x3A08	POSENC_D1AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	D1ADD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x3A0C - 0x3B00	Reserved	Res.																															
0x3B04	POSENC_D0R	D0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B08	POSENC_D1R	D1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B0C	POSENC_VPOSCRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VPOSCRCH[7:0]								VPOSCRCL[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B10	POSENC_VPOS4R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VPOS4[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x3B14	POSENC_VPOS30R	VPOS3[7:0]								VPOS2[7:0]								VPOS1[7:0]								VPOS0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B18	POSENC_VEL20R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VEL2[7:0]								VEL1[7:0]								VEL0[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B1C	POSENC_POS4R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POS4[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x3B20	POSENC_POS30R	POS3[7:0]								POS2[7:0]								POS1[7:0]								POS0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B24 - 0x3FCC	Reserved	Res.																															
0x3FD0	POSENC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

58 Universal serial bus full-speed host/device interface (USB)

This section applies to STM32G55x/56x devices only.

58.1 USB introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB2 bus.

USB suspend/resume are supported, which permits to stop the device clocks for low-power consumption.

58.2 USB main features

- USB specification version 2.0 full-speed compliant
- Supports both Host and Device modes
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint/channel support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support (Device mode only)
- Battery Charging Specification Revision 1.2 support (Device mode only)
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

58.3 USB implementation

Table 671 describes the USB implementation in the devices.

Table 671. STM32G55x/56x USB implementation

USB features(1)	USB
Host mode	X
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	2048 bytes
Dedicated packet buffer memory SRAM access scheme	32 bits
USB 2.0 Link Power Management (LPM) support in device	X

Table 671. STM32G55x/56x USB implementation (continued)

USB features(1)	USB
Battery Charging Detection (BCD) support for device	X
Embedded pull-up resistor on USB_DP line	X

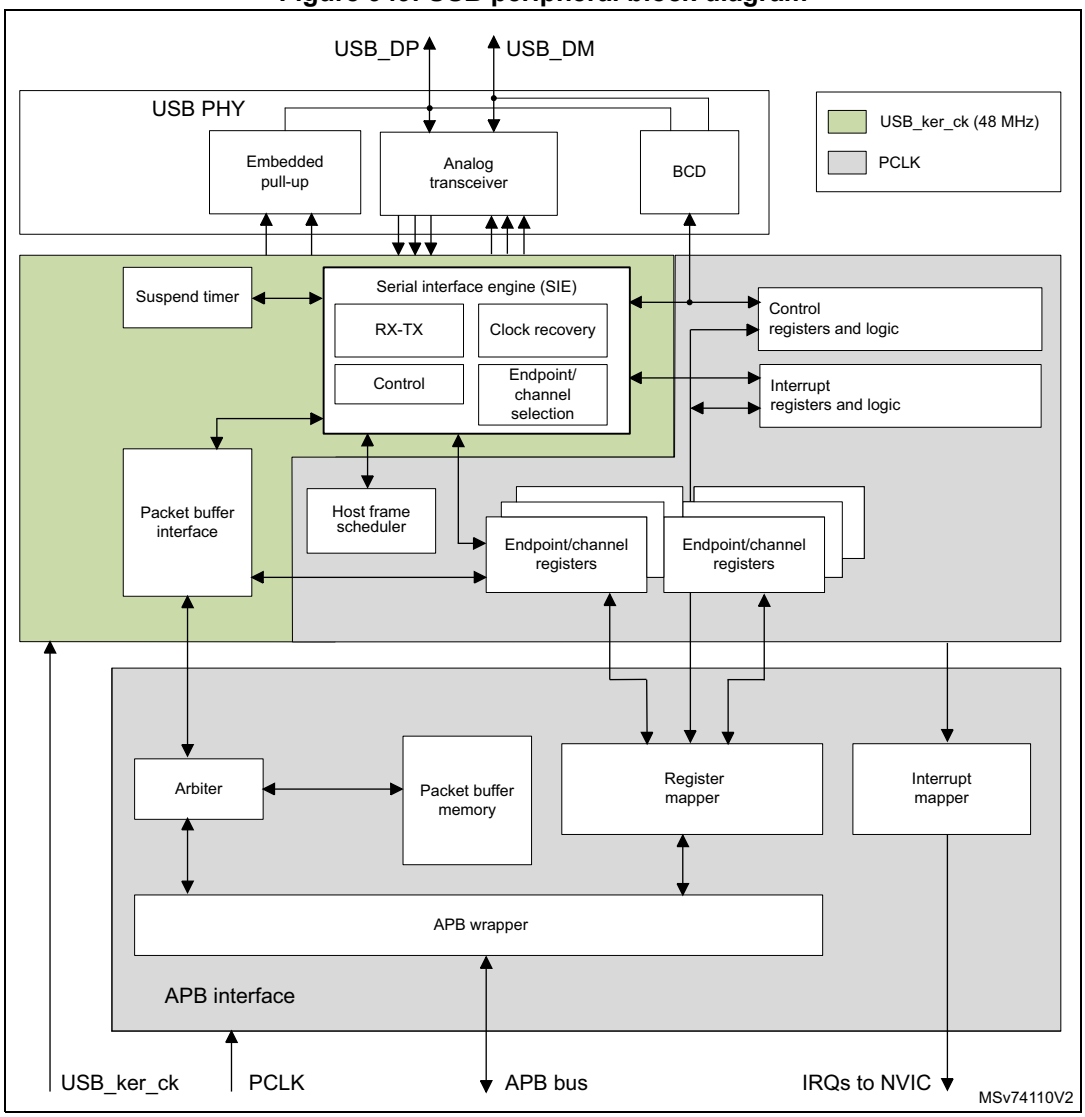
1. X= supported

58.4 USB functional description

58.4.1 USB block diagram

Figure 949 shows the block diagram of the USB peripheral.

Figure 949. USB peripheral block diagram



58.4.2 USB pins and internal signals

Table 672. USB input/output pins

Pin name	Pin type	Description
USB_DP	Digital input/output	D+ line
USB_DM	Digital input/output	D- line

58.4.3 USB reset and clocks

A single reset is present on USB. The RCC allows a reset to be forced by software.

There are two clocks:

- PCLK for the APB bus interface and registers.
- USB_ker_ck (48 MHz) for the main protocol logic including notably the serial interface engine (SIE), see USB_ker_ck clock domain in block diagram.

58.4.4 General description and Device mode functionality

The USB peripheral provides a USB-compliant connection between the function implemented by the microcontroller and an external USB function which can be a host PC but also a USB Device. Data transfer between the external USB host or device and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 2048 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the external USB Host or Device, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint/channel is associated with a buffer description block indicating where the endpoint/channel-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint/channel is configured) takes place. The data buffered by the USB peripheral are loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data have been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint/channel-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint/channel has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which permits to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

A special bit THR512 in register USB_ISTR allows notification of 512 bytes being received in (or transmitted from) the buffer. This bit must be used for long ISO packets (from 512 to 1023 bytes) as it facilitates early start or read/write of data. In this way, the first 512 bytes

can be handled by software while avoiding use of double buffer mode. This bit works when only one ISO endpoint is configured.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to permit the system to immediately restart the normal clock generation and/or support direct clock start/stop.

Host mode and specific functionality

A single bit, HOST, in register USB_CNTR permits Host mode to be activated. Host mode functionality permits the USB to talk to a remote peripheral. Supported functionality is aligned to Device mode and uses the same register structures to manage the buffers. The same number of endpoints can be supported in Host mode, however in Host mode the terminology “channel” is preferred, as each channel is in reality a combination of the connected device and the endpoint on that device. The basic mechanisms for packet transmission and reception are the same as those supported in Device mode.

When operating in Host mode, the USB is in charge of the bus and in order to do this must issue transaction requests corresponding to active periodic and non-periodic endpoints. A host frame scheduler assures efficient use of the frame. Connection to hubs is supported. Connection to low speed devices is supported, both with a direct connection and through a hub.

Double-buffered mode, as previously described in Device mode, is also supported in Host mode, in both bulk and isochronous channels. The THR512 functionality is also supported (but as in Device mode) only for ISO traffic.

Note: Unlike in Device mode, where there is a detection of battery charging capability (in order to facilitate fast charging), there is no integrated support in Host mode to present battery charging capability (CDP or DCP cases in the standard), the host port is always presented as a default standard data port (SDP).

For LPM (link power management) this feature is not supported in Host mode.

58.4.5 Description of USB blocks used in both Device and Host modes

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB physical interface (USB PHY): this block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for battery charging detection (BCD), multiplexed on same USB_DP and USB_DM lines.
- Serial interface engine (SIE): the functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as start of frame (SOF), USB_Reset, data errors etc. and to endpoint

related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.

- Timer: this block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet buffer interface: this block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the endpoint/channel registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
- Endpoint/channel-related registers: each endpoint/channel has an associated register containing the endpoint/channel type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- Control registers: these are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt registers: these contain the interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

*Note: * Endpoint/channel 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB2 bus through an APB2 interface, containing the following blocks:

- Packet memory: this is the local memory that physically contains the packet buffers. It can be used by the packet buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the packet memory is 2048 bytes, structured as 512 words of 32 bits.
- Arbiter: this block accepts memory requests coming from the APB2 bus and from the USB interface. It resolves the conflicts by giving priority to APB2 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB2 transfers of any length are also allowed by this scheme.
- Register mapper: this block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 32-bit wide word set addressed by the APB2.
- APB2 wrapper: this provides an interface to the APB2 for the memory and register. It also maps the whole USB peripheral in the APB2 address space.
- Interrupt mapper: this block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

58.4.6 Description of host frame scheduler (HFS) specific to Host mode

The host frame scheduler is the hardware machine in charge to submit host channel requests on the bus according to the USB priority order and bandwidth access rules.

Host channels are divided in two categories:

- Periodic channels: isochronous and interrupt traffic types. With guaranteed bandwith access.
- Non-periodic channels: bulk and control traffic types. With best effort service.

The host frame scheduler organizes the full-speed frame in 3 sequential windows

- Periodic service window
- Non-periodic service window
- Black security window

At the start of a new frame, the host scheduler:

1. First considers all periodic channels which were active (STAT bits VALID) at the start of frame
2. Executes single round of service of periodic channels, the periodic service window, in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first
3. When the periodic round is finished, HFS closes the periodic service window and stops servicing periodic traffic even if some periodic channel was re-enabled or some new channel was enabled after the SOF.
4. Starts servicing all non-periodic channels which are currently active (STAT bits VALID) in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first.
5. Executes multiple round-robin service cycles of non-periodic channels until almost the end of frame
6. Non-periodic traffic can be requested at any time and is serviced by HFS with best effort latency, with the exception of a black security window at the end of the frame where new injected requests are directly postponed to the next frame to avoid babbles. This is also true for pending transactions which have not been serviced ahead of the security window.

58.5 Programming considerations for Device and Host modes

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

58.5.1 Generic USB Device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and isochronous transfers. Apart from system reset, an action is always initiated by the USB peripheral, driven by one of the USB events described below.

58.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software must perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the USBRST bit in the CNTR register). Clearing the ISTR register removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint/channel must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB bus reset (RST_DCON interrupt) in Device mode

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral does not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the enable function (EF) bit of the USB_DADDR register and initializing the CHEP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RST_DCON interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of the reset sequence which triggered the interrupt.

USB bus reset in Host mode

In Host mode a bus reset is activated by setting the USBRST bit of the USB_CNTR register. It must subsequently be cleared by software once the minimum active reset time from the standard has been respected.

Structure and usage of packet buffers

Each bidirectional endpoint can receive or transmit data over the bus. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request

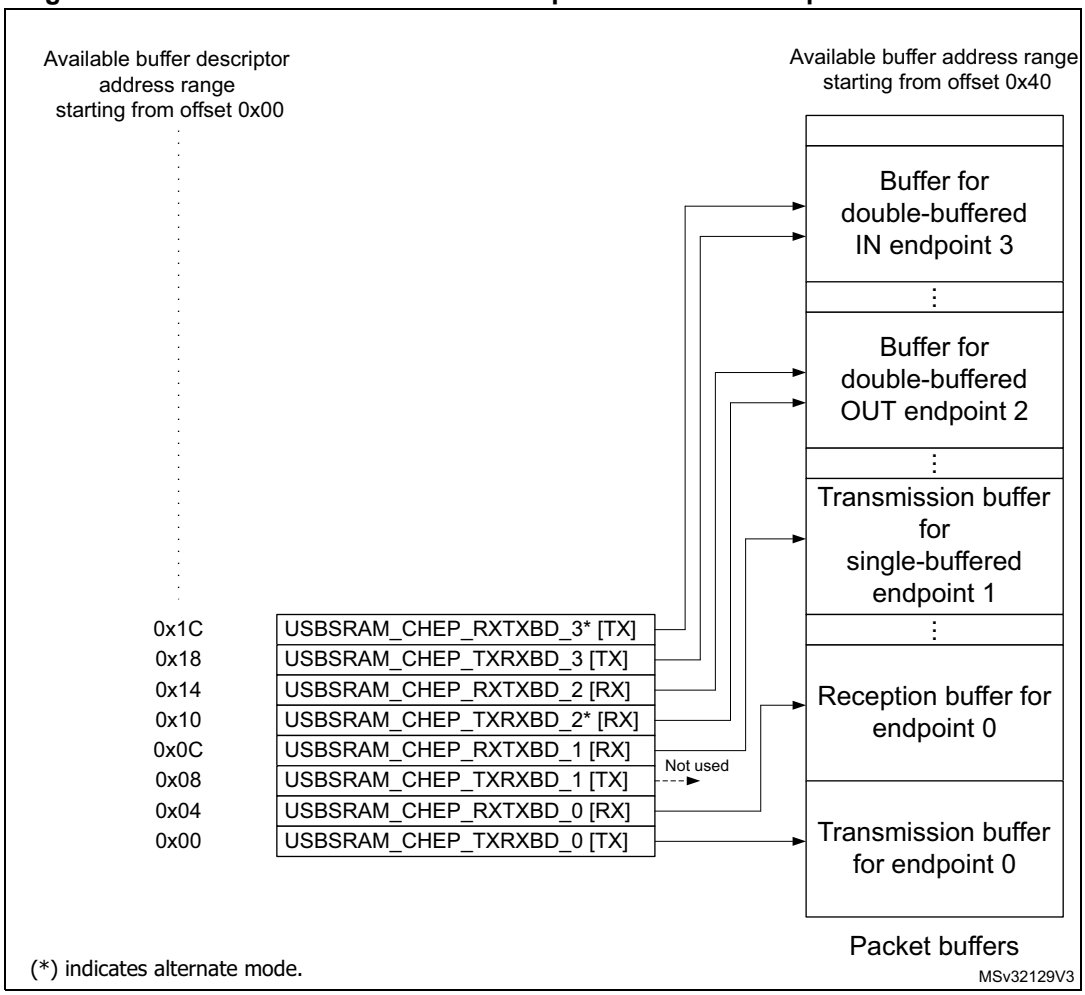
and waits for its acknowledgment. Since the packet buffer memory has also to be accessed by the microcontroller, an arbitration logic takes care of the access conflicts, using half APB2 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both agents can operate as if the packet memory would be a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB2 bus. Different clock configurations are possible where the APB2 clock frequency can be higher or lower than the USB peripheral one.

Note: Due to USB data rate and packet memory interface requirements, the APB2 clock must have a minimum frequency of 12 MHz to avoid data overrun/underrun problems.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory. Each table entry is associated to an endpoint register and it is composed of two 32-bit words so that table start address must always be aligned to an 8-byte boundary. Buffer descriptor table entries are described in *Section 58.7: USBSRAM registers*. If an endpoint is unidirectional and it is neither an isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to *Section 58.5.5: Isochronous transfers in Device mode* and *Section 58.5.3: Double-buffered endpoints and usage in Device mode* respectively). The relationship between buffer description table entries and packet buffer areas is depicted in *Figure 950*.

For Host mode different sections explain the buffer usage model, notably *Section 58.5.6: Isochronous transfers in Host mode* and *Section 58.5.4: Double buffered channels: usage in Host mode*.

Figure 950. Packet buffer areas with examples of buffer description table locations



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral never changes the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data is copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX fields in the CHEP_TXBD_n and CHEP_RXBD_n registers (in SRAM) so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The UTYPE bits in the USB_CHEPnR register must be set according to the endpoint type, eventually using the EPKIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STATTX bits in the USB_CHEPnR register and COUNTn_TX must be initialized. For reception, STATRX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BLSIZE and NUM_BLOCK fields. Unidirectional endpoints, except isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_CHEPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), must not be modified by the application software, as the hardware can

change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

Data transmission in Device mode (IN packets)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of CHEP_TXBD_n (fields ADDRn_TX and COUNTn_TX) inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16-bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (refer to *Structure and usage of packet buffers on page 3114*) and the USB peripheral starts sending a DATA0 or DATA1 PID according to USB_CHEPnR bit DTOGTX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STATTX bits in the USB_CHEPnR register.

The ADDRn_TX field in the internal register CHEP_TXBD_n is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/4 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled, the endpoint is made invalid by setting STATTX = 10 (NAK) and bit VTTX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. Servicing of the VTTX event starts, clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STATTX to 11 (VALID) to re-enable transmission. While the STATTX bits are equal to 10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

Data transmission in Host mode (OUT packets)

Data transmission in Host mode follows the same general principles as Device mode. The main differences are due to the protocol. For example the host initiates the transmission whereas the device responds to the incoming token.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents of an OUT packet are then written to that address in the packet memory and COUNTn_TX must be updated (when necessary) to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATTX must be set to 11 (VALID) in order to trigger the transmit. The transmission is then scheduled by the HFS.

After a successful transmission the CTR interrupt (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the

indicated channel, the STATTX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATTX is now in NAK. In the case of a STALL response, STATTX is in STALL. In this last case, the bus must be reset.

On receiving the ACK receipt by the device, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled.

An error condition is signaled via the bits VTTX and ERR_TX if one of the following conditions occurs:

- No handshake being received in time
- False EOP
- Bit stuffing error
- Invalid handshake PID

Data reception in Device mode (OUT and SETUP packets)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX fields inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX field is stored directly in its internal register ADDR. Internal register COUNT is now reset and the values of BLSIZE and NUM_BLOCK bit fields, which are read within USBSRAM_CHEP_RXBD_n content, are used to initialize BUF_COUNT, an internal 16-bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but the ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STATRX in the USB_CHEPnR register, and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, or up to the last allocated memory location, as defined by BLSIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer

description table entry, leaving unaffected BLSIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_CHEPnR register is updated in the following way: DTOGRX bit is toggled, the endpoint is made invalid by setting STATRX = 10 (NAK) and bit VTRX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. The VTRX event is serviced by first determining the transaction type (SETUP bit in the USB_CHEPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software must set the STATRX bits to 11 (VALID) in the USB_CHEPnR, enabling further transactions. While the STATRX bits are equal to 10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Data reception in Host mode (IN packets)

Data reception in Host mode follows the same general principles as Device mode. The main differences are again due to the protocol. In the device, data can be received or not, depending on readiness after previous operations, whereas the host only requests receive data when it is ready and able to store them.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents received in the data phase response to the IN token packet are then written to that address in the packet memory and COUNTn_TX gets updated by hardware during this process to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATRX must be set to VALID in order to trigger the reception. The reception is then scheduled by the HFS.

After a successful reception the interrupt CTR (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATRX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATRX now is in NAK. In the case of a STALL response, STATRX is in STALL. In this last case, the bus must be reset. During an IN packet an error condition is signaled via the bits VTRX and ERR_RX if one of the following conditions occurs:

- False EOP
- Bit stuffing error
- Wrong CRC

Control transfers in Device mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STATTX and STATRX are set to 10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_CHEPnR register at each VTRX event to distinguish normal

OUT transactions from SETUP ones. A USB Device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction must be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction must be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software changes NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS_OUT (EPKIND in the USB_CHEPnR register) bit must be set, so that an error is generated if a status transaction is performed with non-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STATRX to VALID (to accept a new command) and STATTX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic does not permit a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STATRX bits are set to 01 (STALL) or 10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued VTRX request not yet acknowledged by the application (for example VTRX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the VTRX interrupt.

Control transfers in Host mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only. A control endpoint must set the SETUP bit in the USB_CHEPnR register. The values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 0. Depending on whether it is a control write or control read then STATTX or STATRX are set to 11 (ACTIVE) in order to trigger the control transfer via the host frame scheduler.

On receiving a CTR interrupt the channel (device address and endpoint) can be determined by examining IDN and DIR bits. Devices are expected to NAK every control unless the packet is corrupted in which case they do not acknowledge. The situation is reflected in the value of STATTX.

In the case of an error condition the ERR bit gets set. One possible case is where a CRC error is seen at the device, in this case no ACK is returned to the host. The host sees no ACK and after an appropriate delay this generates a timeout error with ERR_TX set (which can generate an interrupt).

58.5.3 Double-buffered endpoints and usage in Device mode

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it answers with a NAK handshake and the host PC issues the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_CHEPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from 00 (DISABLED): STATRX if the double-buffered bulk endpoint is enabled for reception, STATTX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOGRX (bit 14 of USB_CHEPnR register) for ‘reception’ double-buffered bulk endpoints or DTOGTX (bit 6 of USB_CHEPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral must know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_CHEPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_CHEPnR register bits and DTOG/SW_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.

Table 673. Double-buffering buffer flag definition

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOGTX (USB_CHEPnR bit 6)	DTOGRX (USB_CHEPnR bit 14)
SW_BUF	USB_CHEPnR bit 14	USB_CHEPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 674. Bulk double-buffering memory buffers usage (Device mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (IN)	0	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by performing the two following actions:

- Writing UTYPE bit field at 00 in its USB_CHEPnR register, to define the endpoint as a bulk
- Setting EPKIND bit at 1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains 11 (VALID). However, as the token packet of a new transaction is received, the actual endpoint status is masked as 10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see *Table 674*). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing 1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control takes place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from 11 (VALID) into the STAT bit pair of the related USB_CHEPnR register. In this case, the USB peripheral always uses the programmed endpoint status, regardless of the buffer usage condition.

58.5.4 Double buffered channels: usage in Host mode

In Host mode the underlying transmit and receive methods for double buffered channels are the same as those described for Device mode.

Similar to the Device mode table, a new table below *Table 675: Bulk double-buffering memory buffers usage (Host mode)* shows the programming settings for OUT and IN tokens.

Table 675. Bulk double-buffering memory buffers usage (Host mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (OUT)	0	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (IN)	0	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

58.5.5 Isochronous transfers in Device mode

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID), any other value produces results not compliant to USB standard. Isochronous endpoints implement double-buffering

to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOGRX for ‘reception’ isochronous endpoints, DTOGTX for ‘transmission’ isochronous endpoints, both in the related USB_CHEPnR register) according to *Table 676*.

Table 676. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (IN)	0	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	USBSRAM_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	USBSRAM_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB_CHEPnR registers used to implement isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have isochronous endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for isochronous transfers due to the lack of handshake phase, the endpoint remains always 11 (VALID). CRC errors or buffer-overrun conditions occurring during isochronous OUT transfers are anyway considered as correct transactions and they always trigger a VTRX event. However, CRC errors set the ERR bit in the USB_ISTR register anyway, in order to notify the software of the possible data corruption.

58.5.6 Isochronous transfers in Host mode

From the host point of view isochronous packets are issued or requested one by frame by the host frame scheduler. There is no NAK/ACK protocol and no resend of data or token.

The mechanism is based on a table very similar to that for Device mode. See *Table 677* to understand the relationship between the DTOG bit buffers and the buffer usage.

Table 677. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (OUT)	0	USBSRAM_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.	USBSRAM_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations
	1	USBSRAM_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations	USBSRAM_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.
Receive (IN)	0	USBSRAM_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.	USBSRAM_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.
	1	USBSRAM_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations	USBSRAM_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID),

Just as in Device mode, the mechanism allows automatic toggle of the DTOG bit. Note that in Host mode, at the same time as this toggle, the STATTX or STATRX of the completed buffer is automatically set to DISABLED, permitting the future buffer to be accessed before re-enabling it by setting it to 11 (VALID).

58.5.7 Suspend/resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to 1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the SUSPEN bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set SUSPRDY bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to ensure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (see “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the SUSPRDY bit in USB_CNTR register asynchronously. Even if this event can trigger a WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure must address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear SUSPEN bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to *Table 678*, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the idle bus state; moreover at the end of a reset sequence the RST_DCON bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which must be handled as usual.

Table 678. Resume event detection

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (for example a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the L2RES bit in the USB_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the L2RES bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: The L2RES bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the SUSPEN bit in USB_CNTR register to 1.

Suspend and resume in Host mode

The basics of the suspend and resume mechanism has been described in the previous section.

From the host stand-point, suspend is entered by writing the SUSPEN bit in USB_CNTR. When suspend entry is confirmed, SUSPRDY (also in USB_CNTR) is set.

Once in suspend, and when the application want to resume the bus, this can be done by setting the L2RES bit in USB_CNTR to 1.

Below in *Table 679*, the different actions recommended after a wake-up event are indicated. According to the different line states after a wake-up event, the interpretation of the event and the suggested behavior are shown. Note that, this table here is somewhat expanded when compared to the previously shown device table, as the host may encounter both full speed and low speed devices which use different line states for both suspend and resume.

Table 679. Resume event detection for host

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Not allowed (noise on bus)	Go back in Suspend mode
“10”	Full speed capable device: Not allowed (noise on bus) Low speed device: Device remote wake-up resume	None
“01”	Full speed capable device: Device remote wake-up resume Low speed device: Not allowed (noise on bus)	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

58.6 USB registers

The USB peripheral registers can be divided into the following groups:

- Common registers: interrupt and control registers. These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.
  - USB_CNTR
  - USB_ISTR
  - USB_FNR
  - USB_DADDR
  - USB_LPMCSR
  - USB_BCDR
- Endpoint/channel registers: endpoint/channel configuration and status
  - USB_CHEPnR

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

58.6.1 USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M
rw														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	r	rw	rw

Bit 31 **HOST:** HOST mode

HOST bit selects betweens host or device USB mode of operation. It must be set before enabling the USB peripheral by the function enable bit.

0: USB Device function

1: USB host function

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 **DDISCM**: Device disconnection mask

– Host mode

0: Device disconnection interrupt disabled

1: Device disconnection interrupt enabled

Bit 16 **THR512M**: 512 byte threshold interrupt mask

0: 512 byte threshold interrupt disabled

1: 512 byte threshold interrupt enabled

Bit 15 **CTRM:** Correct transfer interrupt mask

0: Correct transfer (CTR) interrupt disabled.

1: CTR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 **PMAOVRM:** Packet memory area over / underrun interrupt mask

0: PMAOVR interrupt disabled.

1: PMAOVR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 **ERRM:** Error interrupt mask

0: ERR interrupt disabled.

1: ERR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 **WKUPM:** Wake-up interrupt mask

0: WKUP interrupt disabled.

1: WKUP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 11 **SUSPM:** Suspend mode interrupt mask

0: Suspend mode request (SUSP) interrupt disabled.

1: SUSP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 **RST_DCONM:** USB reset request (Device mode) or device connect/disconnect (Host mode) interrupt mask
0: RESET interrupt disabled.
1: RESET interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 **SOFM:** Start of frame interrupt mask
0: SOF interrupt disabled.
1: SOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 **ESOFM:** Expected start of frame interrupt mask
0: Expected start of frame (ESOF) interrupt disabled.
1: ESOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 7 **L1REQM:** LPM L1 state request interrupt mask
0: LPM L1 state request (L1REQ) interrupt disabled.
1: L1REQ interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **L1RES:** L1 remote wake-up / resume driver
– Device mode
Software sets this bit to send a LPM L1 50 µs remote wake-up signaling to the host. After the signaling ends, this bit is cleared by hardware.
0: No effect
1: Send 50 µs remote-wake-up signaling to host

Bit 4 **L2RES:** L2 remote wake-up / resume driver
– Device mode
The microcontroller can set this bit to send remote wake-up signaling to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the host PC is ready to drive the resume sequence up to its end.
– Host mode
Software sets this bit to send resume signaling to the device.
Software clears this bit to send end of resume to device and restart SOF generation.
In the context of remote wake up, this bit is to be set following the WAKEUP interrupt.
0: No effect
1: Send L2 resume signaling to device

Bit 3 **SUSPEN:** Suspend state enable

– Condition: Device mode

Software can set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms. Software can also set this bit when the L1REQ interrupt is received with positive acknowledge sent.

As soon as the suspend state is propagated internally all device activity is stopped, USB clock is gated, USB transceiver is set into low power mode and the SUSPRDY bit is set by hardware. In the case that device application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the microcontroller to stop mode, as in the case of bus powered device application, it must first wait few cycles to see the SUSPRDY = 1 acknowledge the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

– Condition: Host mode

Software can set this bit when host application has nothing scheduled for the next frames and wants to enter long term power saving. When set, it stops immediately SOF generation and any other host activity, gates the USB clock and sets the transceiver in low power mode. If any USB transaction is on-going at the time SUSPEN is set, suspend is entered at the end of the current transaction.

As soon as suspend state is propagated internally and gets effective the SUSPRDY bit is set. In the case that host application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the micro-controller to STOP mode, it must first wait few cycles to see SUSPRDY=1 acknowledge to the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

Bit 2 **SUSPRDY:** Suspend state effective

This bit is set by hardware as soon as the suspend state entered through the SUSPEN control gets internally effective. In this state USB activity is suspended, USB clock is gated, transceiver is set in low power mode by disabling the differential receiver. Only asynchronous wake-up logic and single ended receiver is kept alive to detect remote wake-up or resume events.

Software must poll this bit to confirm it to be set before any STOP mode entry.

This bit is cleared by hardware simultaneously to the WAKEUP flag being set.

0: Normal operation

1: Suspend state

Bit 1 **PDWN:** Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.
0: Exit power down
1: Enter power down mode

Bit 0 **USBRST:** USB Reset

– Condition: Device mode

Software can set this bit to reset the USB core, exactly as it happens when receiving a RESET signaling on the USB.The USB peripheral, in response to a RESET, resets its internal protocol state machine. Reception and transmission are disabled until the RST_DCON bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RST_DCON interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.
0: No effect
1: USB core is under reset

– Condition: Host mode

Software sets this bit to drive USB reset state on the bus and initialize the device. USB reset terminates as soon as this bit is cleared by software.
0: No effect
1: USB reset driven

58.6.2 USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

This register contains the status of all the interrupt sources permitting application software to determine which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, performs all necessary actions, and finally it clears the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint/channel transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint/channel successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint/channel dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_CHEPnR register (the CTR bit is actually a read only bit). For endpoint-/channel-related interrupts, the software can use the direction of transaction (DIR) and IDN read-only bits to identify which endpoint/channel made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with 0 (these bits can only be cleared by software). Read-modify-write cycles must be avoided because between the read and the write operations another bit can be set by the hardware and the next write clears it before the device has the time to service the event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LS_ DCON	DCON_ STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR 512
	r	r												rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RST_ DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LS_DCON**: Low speed device connected

– Host mode:

This bit is set by hardware when an LS device connection is detected. Device connection is signaled after LS J-state is sampled for 22 consecutive cycles of the USB clock (48 MHz) from the unconnected state.

Bit 29 **DCON_STAT**: Device connection status

– Host mode:

This bit contains information about device connection status. It is set by hardware when a LS/FS device is attached to the host while it is reset when the device is disconnected.

0: No device connected

1: FS or LS device connected to the host

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **DDISC**: Device connection

– Host mode

This bit is set when a device connection is detected. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 16 **THR512**: 512 byte threshold interrupt

This bit is set to 1 by the hardware when 512 bytes have been transmitted or received during isochronous transfers. This bit is read/write but only 0 can be written and writing 1 has no effect. Note that no information is available to indicate the associated channel/endpoint, however in practice only one ISO endpoint/channel with such large packets can be supported, so that channel.

Bit 15 **CTR**: Completed transfer in host mode

This bit is set by the hardware to indicate that an endpoint/channel has successfully completed a transaction; using DIR and IDN bits software can determine which endpoint/channel requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR:** Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host retries the transaction. The PMAOVR interrupt must never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 13 **ERR:** Error

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic redundancy check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (for example loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 12 **WKUP:** Wake-up

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the SUSPRDY bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (for example wake-up unit) about the start of the resume process. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 11 **SUSP:** Suspend mode request

– Device mode

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (SUSPEN=1) until the end of resume sequence. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 10 **RST_DCON:** USB reset request (Device mode) or device connect/disconnect (Host mode)

– Device mode

This bit is set by hardware when an USB reset is released by the host and the bus returns to idle. USB reset state is internally detected after the sampling of 60 consecutive SE0 cycles.

– Host mode

This bit is set by hardware when device connection or device disconnection is detected. Device connection is signaled after J state is sampled for 22 cycles consecutively from unconnected state. Device disconnection is signaled after SE0 state is seen for 22 bit times consecutively from connected state.

Bit 9 **SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine can monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this can be useful for isochronous applications). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 8 **ESOF:** Expected start of frame

– Device mode

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the suspend timer issues this interrupt. If three consecutive ESOF interrupts are generated (for example three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the suspend timer is not yet locked. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 7 **L1REQ:** LPM L1 state request

– Device mode

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **DIR:** Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.
If DIR bit = 0, VTTX bit is set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).
If DIR bit = 1, VTRX bit or both VTTX/VTRX are set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed. This information can be used by the application software to access the USB_CHEPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **IDN[3:0]:** Device Endpoint / host channel identification number

These bits are written by the hardware according to the host channel or device endpoint number, which generated the interrupt request. If several endpoint/channel transactions are pending, the hardware writes the identification number related to the endpoint/channel having the highest priority defined in the following way: two levels are defined, in order of priority: isochronous and double-buffered bulk channels/endpoints are considered first and then the others are examined. If more than one endpoint/channel from the same set is requesting an interrupt, the IDN bits in USB_ISTR register are assigned according to the lowest requesting register, CHEP0R having the highest priority followed by CHEP1R and so on. The application software can assign a register to each endpoint/channel according to this priority scheme, so as to order the concurring endpoint/channel requests in a suitable way. These bits are read only.

58.6.3 USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0000 0XXX (where X is undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RXDP:** Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 14 **RXDM:** Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 13 **LCK:** Locked

– Device mode

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]:** Lost SOF

– Device mode

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]:** Frame number

This bit field contains the 11-bit frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

58.6.4 USB Device address (USB_DADDR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **EF:** Enable function

This bit is set by the software to enable the USB Device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at 0 no transactions are handled, irrespective of the settings of USB_CHEPnR registers.

Bits 6:0 **ADD[6:0]:** Device address

– Device mode

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the endpoint/channel address (EA) field in the associated USB_CHEPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

– Host mode

These bits contain the address transmitted with the LPM transaction

58.6.5 USB LPM control and status register (USB_LPMCSR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REM WAKE	Res.	LPM ACK	LPM EN
								r	r	r	r	r		rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **BESL[3:0]:** BESL value

– Device mode

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REMWAKE:** bRemoteWake value

– Device mode

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LPMACK:** LPM token acknowledge enable

– Device mode:

0: the valid LPM token is NYET.
1: the valid LPM token is ACK.
The NYET/ACK is returned only on a successful LPM transaction:
No errors in both the EXT token and the LPM token (else ERROR)
A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 **LPMEN:** LPM support enable

– Device mode

This bit is set by the software to enable the LPM support within the USB Device. If this bit is at 0 no LPM transactions are handled.

58.6.6 USB battery charging detector (USB_BCDR)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	Res.	SDEN	PDEN	Res.	BCDEN
rw								r	r	r		rw	rw		rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DPPU_DPD:** DP pull-up / DPDM pull-down

– Device mode

This bit is set by software to enable the embedded pull-up on DP line. Clearing it to 0 can be used to signal disconnect to the host when needed by the user software.

– Host mode

This bit is set by software to enable the embedded pull-down on DP and DM lines.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET:** DM pull-up detection status

– Device mode

This bit is active only during PD and gives the result of comparison between DM voltage level and V_{LGC} threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).

1: PS2 port or proprietary charger detected.

Bit 6 **SDET:** Secondary detection (SD) status

– Device mode

This bit gives the result of SD.

0: CDP detected.

1: DCP detected.

Bit 5 **PDET:** Primary detection (PD) status

– Device mode

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 Reserved, must be kept at reset value.

Bit 3 **SDEN:** Secondary detection (SD) mode enable

– Device mode

This bit is set by the software to put the BCD into SD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly.

Bit 2 **PDEN:** Primary detection (PD) mode enable
– Device mode
This bit is set by the software to put the BCD into PD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCDEN:** Battery charging detector (BCD) enable
– Device mode
This bit is set by the software to enable the BCD support within the USB Device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to 0 in order to allow the normal USB operation.

58.6.7 USB endpoint/channel n register (USB_CHEPnR)

Address offset: 0x00 + 0x4 * n, (n = 0 to 7)

Reset value: 0x0000 0000

The USB peripheral supports up to 8 bidirectional endpoints or host channels. Each USB Device must support a control endpoint/channel whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint/channel number value. For each endpoint, an USB_CHEPnR register is available to store the endpoint/channel specific information.

They are also reset when an USB reset is received from the USB bus or forced through bit USBRST in the CTLR register, except the VTRX and VTTX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint/channel has its USB_CHEPnR register where *n* is the endpoint/channel identifier.

Read-modify-write cycles on these registers must be avoided because between the read and the write operations some bits can be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	THREE_ERR_RX[1:0]		THREE_ERR_TX[1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	rc_w0	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VTRX	DTOGRX	STATRX[1:0]		SETUP	UTYPE[1:0]		EP KIND	VTTX	DTOGTX	STATTX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **THREE_ERR_RX[1:0]:** Three errors for an IN transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an IN transaction. THREE_ERR_RX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bits 28:27 **THREE_ERR_TX[1:0]:** Three errors for an OUT or SETUP transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an OUT transaction. THREE_ERR_TX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bit 26 **ERR_RX:** Received error for an IN transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an IN transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 25 **ERR_TX:** Received error for an OUT/SETUP transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an OUT or SETUP transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 24 **LS_EP:** Low speed endpoint – host with HUB only

– Host mode

This bit is set by the software to send an LS transaction to the corresponding endpoint.

0: Full speed endpoint

1: Low speed endpoint

Bit 23 **NAK:**

– Host mode

This bit is set by the hardware when a device responds with a NAK. Software can use this bit to monitor the number of NAKs received from a device.

Bits 22:16 **DEVADDR[6:0]:**

– Host mode

Device address assigned to the endpoint during the enumeration process.

Bit 15 **VTRX:** USB valid transaction received

– Device mode

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only 0 can be written, writing 1 has no effect.

– Host mode

This bit is set by the hardware when an IN transaction is successfully completed on this channel. The software can only clear this bit. If the CTRM bit in USB_CNTR register is set a generic interrupt condition is generated together with the channel related flag, which is always activated.

- A transaction ended with a NAK sets this bit and NAK answer is reported to application reading the NAK state from the STATRX field of this register. One NAKed transaction keeps pending and is automatically retried by the host at the next frame, or the host can immediately retry by resetting STATRX state to VALID.

- A transaction ended by STALL handshake sets this bit and the STALL answer is reported to application reading the STALL state from the STATRX field of this register. Host application must consequently disable the channel and re-enumerate.

- A transaction ended with ACK handshake sets this bit

If double buffering is disabled, ACK answer is reported by application reading the DISABLE state from the STATRX field of this register. Host application must read received data from USBRAM and re-arm the channel by writing VALID to the STATRX field of this register.

If double buffering is enabled, ACK answer is reported by application reading VALID state from the STATRX field of this register. Host application must read received data from USBRAM and toggle the DTOGTX bit of this register.

- A transaction ended with error sets this bit.

Errors can be seen via the bits ERR_RX (host mode only).

This bit is read/write but only 0 can be written, writing 1 has no effect.

Bit 14 **DTOGRX:** Data Toggle, for reception transfers

If the endpoint/channel is not isochronous, this bit contains the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID received from host (in device mode), while it sets this bit to 1 when SETUP transaction is acknowledged by device (in host mode).
If the endpoint/channel is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to *Section 58.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used only to support packet buffer swapping for data transmission since no data toggling is used for this kind of channels/endpoints and only DATA0 packet are transmitted (Refer to *Section 58.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGRX remains unchanged, while writing 1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STATRX[1:0]:** Status bits, for reception transfers

– Device mode

These bits contain information about the endpoint status, which are listed in *Table 680: Reception status encoding on page 3147*. These bits can be toggled by software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATRX bits to NAK when a correct transfer has occurred (VTRX = 1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledges a new transaction.

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to *Section 58.5.3: Double-buffered endpoints and usage in Device mode*).

If the endpoint is defined as isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STATRX bits to ‘STALL’ or ‘NAK’ for an isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

These bits are the host application controls to start, retry, or abort host transactions driven by the channel.

These bits also contain information about the device answer to the last IN channel transaction and report the current status of the channel according to the following STATRX table of states:

- DISABLE

  DISABLE value is reported in case of ACK acknowledge is received on a single-buffer channel. When in DISABLE state the channel is unused or not active waiting for application to restart it by writing VALID. Application can reset a VALID channel to DISABLE to abort a transaction. In this case the transaction is immediately removed from the host execution list. If the aborted transaction was already under execution it is regularly terminated on the USB but the relative VTRX interrupt is not generated.

- VALID

  A host channel is actively trying to submit USB transaction to device only when in VALID state.VALID state can be set by software or automatically by hardware on a NAKED channel at the start of a new frame. When set to VALID, an host channel enters the host execution queue and waits permission from the host frame scheduler to submit its configured transaction.

  VALID value is also reported in case of ACK acknowledge is received on a double-buffered channel. In this case the channel remains active on the alternate buffer while application needs to read the current buffer and toggle DTOGTX. In case software is late in reading and the alternate buffer is not ready, the host channel is automatically suspended transparently to the application. The suspended double buffered channel is re-activated as soon as delay is recovered and DTOGTX is toggled.

- NAK

  NAK value is reported in case of NAK acknowledge received. When in NAK state the channel is suspended and does not try to transmit. NAK state is moved to VALID by hardware at the start of the next frame, or software can change it to immediately retry transmission by writing it to VALID, or can disable it and abort the transaction by writing DISABLE

- STALL

  STALL value is reported in case of STALL acknowledge received. When in STALL state the channel behaves as disabled. Application must not retry transmission but reset the USB and re-enumerate.

Bit 11 **SETUP:** Setup transaction completed

– Device mode

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (VTRX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while VTRX bit is at 1; its state changes when VTRX is at 0. This bit is read-only.

– Host mode

This bit is set by the software to send a SETUP transaction on a control endpoint. This bit changes its value only for control endpoints. It is cleared by hardware when the SETUP transaction is acknowledged and VTTX interrupt generated.

Bits 10:9 **UTYPE[1:0]:** USB type of transaction

These bits configure the behavior of this endpoint/channel as described in *Table 681: Endpoint/channel type encoding*. Channel0/Endpoint0 must always be a control endpoint/channel and each USB function must have at least one control endpoint/channel which has address 0, but there can be other control channels/endpoints if required. Only control channels/endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint/channel is defined as NAK, the USB peripheral does not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint/channel is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint/channel is a control one. Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EPKIND configuration bit.
The usage of isochronous channels/endpoints is explained in *Section 58.5.5: Isochronous transfers in Device mode*

Bit 8 **EPKIND:** endpoint/channel kind

The meaning of this bit depends on the endpoint/channel type configured by the UTYPE bits. *Table 682* summarizes the different meanings.
DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 58.5.3: Double-buffered endpoints and usage in Device mode*.
STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit can be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **VTTX:** Valid USB transaction transmitted

– Device mode

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.
A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
This bit is read/write but only 0 can be written.

– Host mode

Same as VTRX behavior but for USB OUT and SETUP transactions.

Bit 6 **DTOGTX:** Data toggle, for transmission transfers

If the endpoint/channel is non-isochronous, this bit contains the required value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint/channel is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint (in device mode) or when a SETUP transaction is acknowledged by the device (in host mode).
If the endpoint/channel is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to *Section 58.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (refer to *Section 58.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint/channel is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGTX remains unchanged, while writing 1 makes the bit value to toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STATTX[1:0]:** Status bits, for transmission transfers

– Device mode

These bits contain the information about the endpoint status, listed in *Table 683*. These bits can be toggled by the software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATTX bits to NAK, when a correct transfer has occurred (VTTX = 1) corresponding to a IN or SETUP (control only) transaction addressed to this channel/endpoint. It then waits for the software to prepare the next set of data to be transmitted.
Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to *Section 58.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint is defined as isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the channel/endpoint/channel after a successful transaction. If the software sets the STATTX bits to ‘STALL’ or ‘NAK’ for an isochronous channel/endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

The STATTX bits contain the information about the channel status. Refer to *Table 683* for the full descriptions (“Host mode” descriptions). Whereas in Device mode, these bits contain the status that are given out on the following transaction, in Host mode they capture the status last received from the device. If a NAK is received, STATTX contains the value indicating NAK.

Bits 3:0 **EA[3:0]:** endpoint/channel address

– Device mode

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

– Host mode

Software must write in this field the 4-bit address used to identify the channel addressed by the host transaction.

Table 680. Reception status encoding

STATRX[1:0]	Meaning
00	**DISABLED:** all reception requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all reception requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.
10	**NAK**: Device mode: the endpoint is NAKed and all reception requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the reception request.
11	**VALID**: this endpoint/channel is enabled for reception.

Table 681. Endpoint/channel type encoding

UTYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 682. Endpoint/channel kind meaning

UTYPE[1:0]		EPKIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	SBUF_ISO: This bit is set by the software to enable the single-buffering feature for isochronous endpoint
11	INTERRUPT	Not used

Table 683. Transmission status encoding

STATTX[1:0]	Meaning
00	**DISABLED:** all transmission requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all transmission requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.

Table 683. Transmission status encoding (continued)

STATTX[1:0]	Meaning
10	**NAK**: Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the transmission request.
11	**VALID**: this endpoint/channel is enabled for transmission.

58.6.8 USB register map

The table below provides the USB register map and reset values.

Table 684. USB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USB_CHEP0R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USB_CHEP1R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USB_CHEP2R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USB_CHEP3R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USB_CHEP4R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USB_CHEP5R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 684. USB register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	USB_CHEP6R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	USB_CHEP7R	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	USB_CNTR	HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
	Reset value	0														0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	1
0x44	USB_ISTR	Res.	LS_DCON	DCON_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR512	CTR	PMAOVR	ERR	WKUP	SUSP	RST_DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
	Reset value		0	0												0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x48	USB_FNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	USB_DADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x54	USB_LPMCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REMWAKE	Res.	LPMACK	LPMEN
	Reset value																									0	0	0	0	0		0	0
0x58	USB_BCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	Res.	SDEN	PDEN	Res.	BCDEN
	Reset value																	0								0	0	0		0	0		0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

58.7 USBSRAM registers

Note: The buffer descriptor table is located inside the packet buffer memory in the separate "USBSRAM" address space.

Although the buffer descriptor table is located inside the packet buffer memory ("USBSRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USBSRAM base address. The buffer descriptor table entry associated with the USB_CHEPnR registers is described below. The memory must be addressed using Word (32-bit) accesses.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in *Structure and usage of packet buffers on page 3114*.

58.7.1 Channel/endpoint transmit buffer descriptor n (USBSRAM_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program the use of double buffering working in receive mode (otherwise refer to following register description)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

58.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USBSRAM_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs the use of double buffering and activates receive buffers (otherwise refer to previous register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during

the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

– If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 685*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

Table 685. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
0 (00000)	Not allowed	32 bytes
1 (00001)	2 bytes	64 bytes
2 (00010)	4 bytes	96 bytes
3 (00011)	6 bytes	128 bytes
...	...	...
14 (01110)	28 bytes	480 bytes
15 (01111)	30 bytes	512 bytes
16 (10000)	32 bytes	544 bytes
...	...	...

Table 685. Definition of allocated buffer memory (continued)

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
29 (11101)	58 bytes	...
30 (11110)	60 bytes	992 bytes
31 (11111)	62 bytes	1023 bytes

58.7.3 Channel/endpoint receive buffer descriptor n (USBSRAM_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program use of double buffering in the transmit mode (otherwise refer to following register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

– If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 685*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

58.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USBSRAM_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs use of double buffering and activates transmit buffers (otherwise refer to previous register description).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

58.7.5 USBSRAM register map

The table below provides the USB register map and reset values.

Table 686. USBSRAM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USBSRAM_CHEP_TXRXBD_0	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x00	USBSRAM_CHEP_TXRXBD_0 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	USBSRAM_CHEP_RXTXBD_0	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	USBSRAM_CHEP_RXTXBD_0 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	USBSRAM_CHEP_TXRXBD_1	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	USBSRAM_CHEP_TXRXBD_1 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	USBSRAM_CHEP_RXTXBD_1	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	USBSRAM_CHEP_RXTXBD_1 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
. . . .	. . .	. . .																															
0x38	USBSRAM_CHEP_TXRXBD_7	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x38	USBSRAM_CHEP_TXRXBD_7 [alternate]	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	USBSRAM_CHEP_RXTXBD_7	BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	USBSRAM_CHEP_RXTXBD_7 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

59 USB Type-C®/USB Power Delivery interface (UCPD)

This section only applies to STM32G55x/56x devices.

59.1 UCPD introduction

The USB Type-C/USB Power Delivery interface complies with:

- Universal Serial Bus Type-C Cable and Connector Specification: release 2.5, March 2026
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.2, version 1.2, August 2026

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

59.2 UCPD main features

- Compliance with USB Type-C specification release 2.5
- Compliance with USB Power Delivery specifications revision 2.0 and 3.2
  - Enabling advanced applications such as PPS (programmable power supply)
- Stop mode low-power operation support
- Built-in analog PHY
  - USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
  - “Dead battery” Rd support
  - USB Power Delivery message transmission and reception
  - FRS (fast role swap) Rx support
- Digital controller
  - BMC (bi-phase mark coding) encode and decode
  - 4b5b encode and decode
  - USB Type-C level detection with de-bounce, generating interrupts
  - FRS signaling
  - FRS detection, generating an interrupt
  - DMA-compatible byte-level interface for USB Power Delivery payload, generating interrupts
  - USB Power Delivery clock pre-scaler / dividers
  - CRC generation/checking
  - Support of ordered sets, with a programmable ordered set mask at receive
  - Clock recovery from incoming Rx stream

59.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.

In the following table, “X” denotes *supported*, and “-” *not supported*.

Table 687. UCPD implementation

UCPD feature	UCPD1
Dead battery support via UCPDx_DBCC1 and UCPDx_DBCC2 external signals	X
UCPDx_FRSTX as alternate function pin	X
Fully automatic trimming	-(1)
USB PD receiver hardware filter control	X
Discrete-component PHY support	-

1. Apply software trimming as described in *Section 59.5.5: UCPD software trimming*.

The following table gives the memory locations of trim data stored in the nonvolatile memory, to use in the software trimming procedure described in *Section 59.5.5: UCPD software trimming*.

Table 688. UCPD software trim data

Name	Nonvolatile memory location	
	Address	Bits
3A0_CC1[3:0]	0x4002 242C	7:4
3A0_CC2[3:0]	0x4002 242C	15:12
1A5_CC1[3:0]	0x08FF F844	3:0
1A5_CC2[3:0]	0x08FF F844	19:16
Rd_CC1[3:0]	0x4002 242C	3:0
Rd_CC2[3:0]	0x4002 242C	11:8

59.4 UCPD functional description

The UCPD peripheral provides hardware support for the USB Power Delivery control interface specification, using I/Os specifically designed for that purpose.

The built-in PHY directly detects Type-C voltage levels, supports Power Delivery BIST carrier mode 2 (Tx only), BIST test data (Tx and Rx), and Power Delivery Rx FRS signaling.

For Power Delivery FRS Tx signaling, the device can be configured to control, through UCPD_FRSTX pin (alternate function), external NMOS transistors that ensure low-resistance pull-down on CC lines.

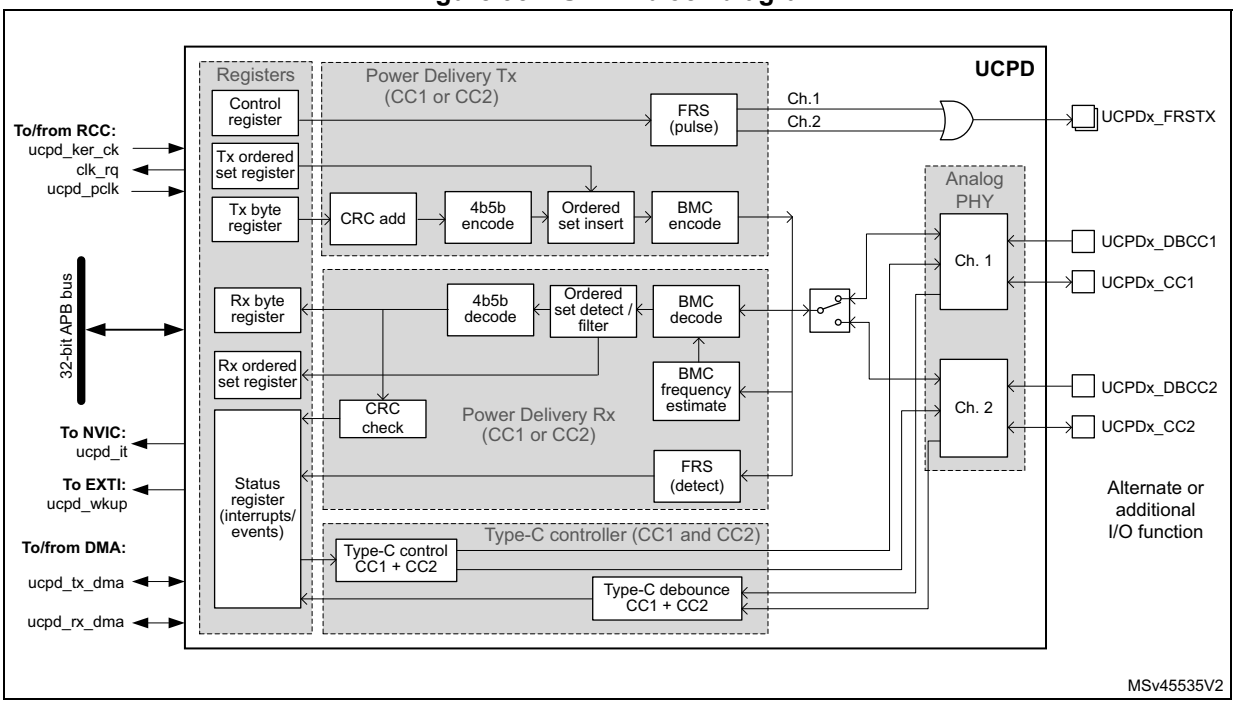
The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes “Hard Reset”.

The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns (allows for only three out of four K-codes being correctly received, as defined by the standard).

In Stop mode, the peripheral maintains the ability to detect incoming USB Power Delivery messages and FRS signaling, which allows low-power operation.

59.4.1 UCPD block diagram

Figure 951. UCPD block diagram



The following table lists external signals (alternate or additional I/O functions).

Table 689. UCPD signals on pins

Pin name	Signal type	Description
UCPDx_FRSTX	Output	USB Type-C fast role swap (FRS) signaling control, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the active CC line. A typical application has two such transistors (one per CC line) and reserves a separate I/O to drive either NMOS. Initially, the I/Os are configured as low-driving GPIOs. Upon detecting, through the Type-C state machine, the orientation of the cable attached, which determines the active CC line, the I/O of the active CC line must be set to its UCPDx_FRSTX alternate function and the I/O of the inactive CC line as low-driving GPIO.
UCPDx_CC1	Input/output	USB Type-C configuration control line 1, to be routed to the USB Type-C connector CC1 terminal.

Table 689. UCPD signals on pins (continued)

Pin name	Signal type	Description
UCPDx_CC2	Input/output	USB Type-C configuration control line 2, to be routed to the USB Type-C connector CC2 terminal.
UCPDx_DBCC1	Input	USB Type-C configuration control line 1 dead battery signal, to be routed to the USB Type-C connector CC1 terminal if dead battery support is required.
UCPDx_DBCC2	Input	USB Type-C configuration control line 2 dead battery signal, to be routed to the USB Type-C connector CC2 terminal if dead battery support is required.

The following table lists key internal signals.

Table 690. UCPD internal signals

Internal signal name	Signal type	Description
ucpd_pclk	Input	APB clock for registers
ucpd_ker_ck	Input	Kernel clock
ucpd_tx_dma	Input/Output	Rx DMA acknowledge / request
ucpd_rx_dma	Input/Output	Tx DMA acknowledge / request
ucpd_it	Output	Interrupt request (all interrupts OR-ed) connected to NVIC
ucpd_wkup	Output	Wake-up request connected to EXTI
clk_rq	Output	Clock request connected to RCC

59.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

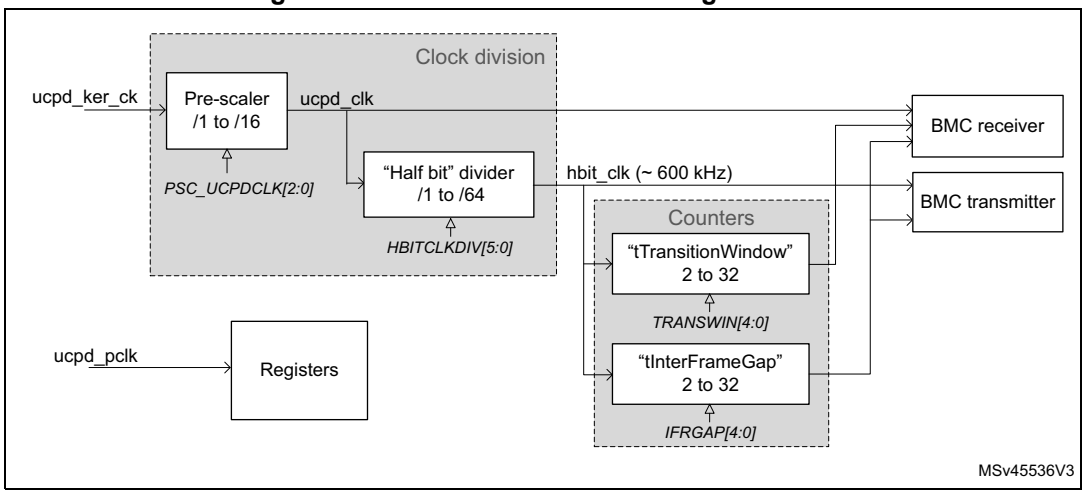
The register section is clocked with the APB clock (ucpd_pclk).

The main functional part of the transmitter is clocked with ucpd_clk clock, pre-scaled from the ucpd_ker_ck clock according to the PSC_UCPDCLK[2:0] bitfield of the UCPD_CFGR1 register. The main functional part of the receiver is clocked with the ucpd_rx_clk recovered from the incoming bitstream.

The receiver is designed to work in the clock frequency range from 6 to 18 MHz. However, the optimum performance is ensured in the range from 6 to 12 MHz.

The following diagram shows the clocking and timing elements of the UCPD peripheral.

Figure 952. Clock division and timing elements



Refer to the USB PD specification in order to set appropriate delays. For *tTransitionWindow* and especially for *tInterFrameGap*, the clock frequency uncertainty must be taken into account so as to respect specified timings in all cases.

59.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification.

On the transmitter side its main function is to form packets according to the defined packet format including generally:

- preamble
- start of packet (SOP, ordered set)
- payload header
- payload data
- cyclic redundancy check (CRC) information
- end of packet (EOP)

Before going on the CC line, the data stream is BMC-encoded, respecting specified timing restrictions.

On the receive side, the principle task is to:

- extract start of packet (SOP, ordered set) information
- extract payload header
- extract payload data
- receive and check CRC
- receive end of packet (EOP)

The receive is basically a reverse of the transmit process, thus starting with BMC data stream decoding.

Symbol encoding

Apart from the preamble all symbols are encoded with a 4b5b scheme according to the specification shown in the following table.

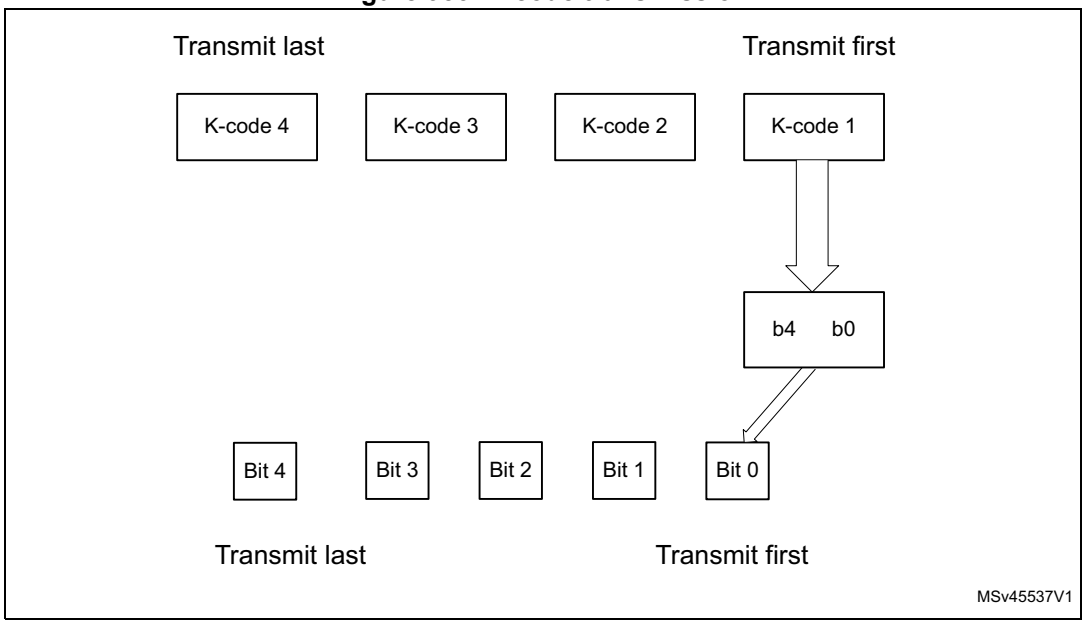
Table 691. 4b5b symbol encoding table

Name	4b	5b	Symbol description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
Sync-1	K-code	11000	Startsynch #1
Sync-2	K-code	10001	Startsynch #2
RST-1	K-code	00111	Hard Reset #1
RST-2	K-code	11001	Hard Reset #2
EOP	K-code	01101	EOP
Reserved	Error	00000	Do Not Use
Reserved	Error	00001	Do Not Use
Reserved	Error	00010	Do Not Use
Reserved	Error	00011	Do Not Use
Reserved	Error	00100	Do Not Use
Reserved	Error	00101	Do Not Use
Sync-3	K-code	00110	Startsynch #3
Reserved	Error	01000	Do Not Use
Reserved	Error	01100	Do Not Use
Reserved	Error	10000	Do Not Use
Reserved	Error	11111	Do Not Use

Ordered sets

An ordered set consists of four K-codes as shown in the following figure.

Figure 953. K-code transmission



The following table lists the defined ordered sets, including all possible SOP* sequences.

At the physical layer, the Hard Reset has higher priority than the other ordered sets so it can interrupt an ongoing Tx message.

Table 692. Ordered sets

Ordered set name	K-code #1	K-code #2	K-code #3	K-code #4
SOP	Sync-1	Sync-1	Sync-1	Sync-2
SOP’	Sync-1	Sync-1	Sync-3	Sync-3
SOP’’	Sync-1	Sync-3	Sync-1	Sync-3
Hard Reset	RST-1	RST-1	RST-1	RST-2
Cable Reset	RST-1	Sync-1	RST-1	Sync-3
SOP’_Debug	Sync-1	RST-2	RST-2	Sync-3
SOP’’_Debug	Sync-1	RST-2	Sync-3	Sync-2

On reception, the physical layer must accept ordered sets with any combination of three correct K-codes out of four, as shown in the following table:

Table 693. Validation of ordered sets

Status	1st code	2nd code	3rd code	4th code
Valid	Corrupt	K-code	K-code	K-code
Valid	K-code	Corrupt	K-code	K-code

Table 693. Validation of ordered sets (continued)

Status	1st code	2nd code	3rd code	4th code
Valid	K-code	K-code	Corrupt	K-code
Valid	K-code	K-code	K-code	Corrupt
Valid (perfect)	K-code	K-code	K-code	K-code
Not valid (example)	K-code	Corrupt	K-code	Corrupt

Bit ordering at transmission

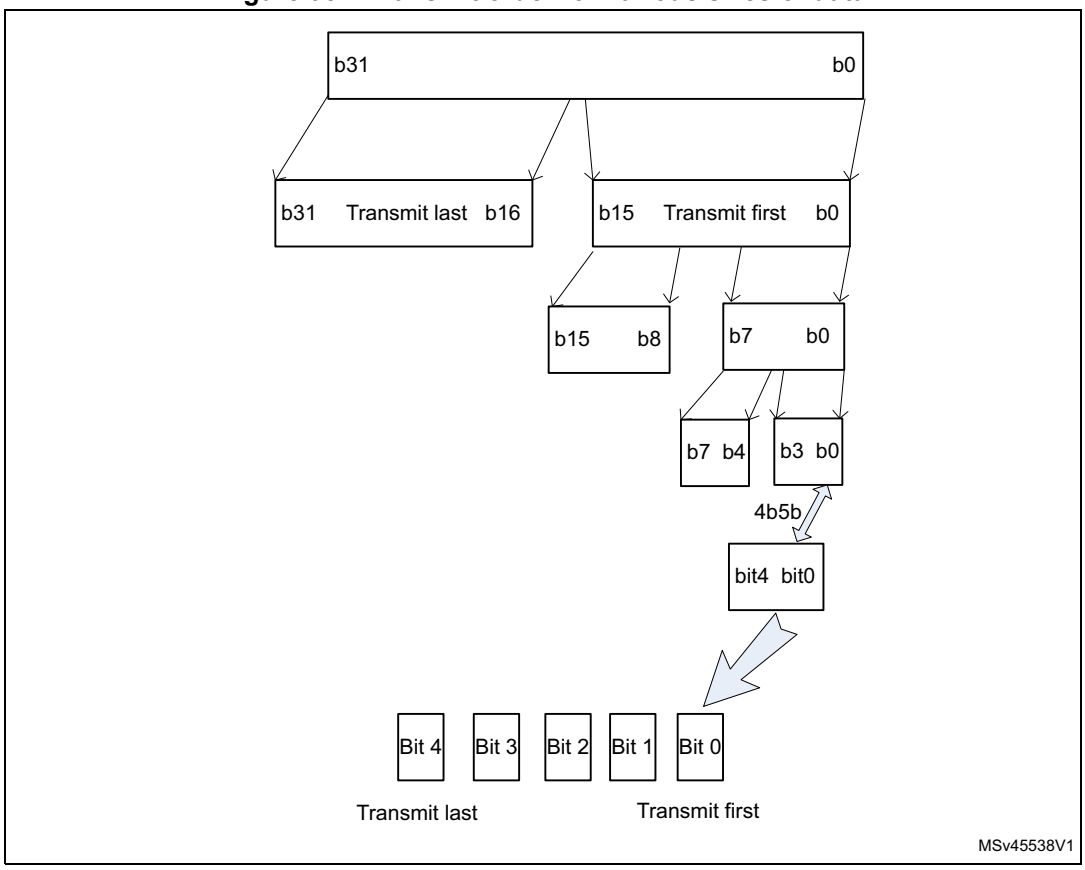
Allowed transmission data units / data sizes are in the following table.

Table 694. Data size

Data unit	Non-encoded	Encoded
Byte	8-bits	10-bits
Word	16-bits	20-bits
DWord	32-bits	40-bits

The bit transmission order is shown in the following figure.

Figure 954. Transmit order for various sizes of data

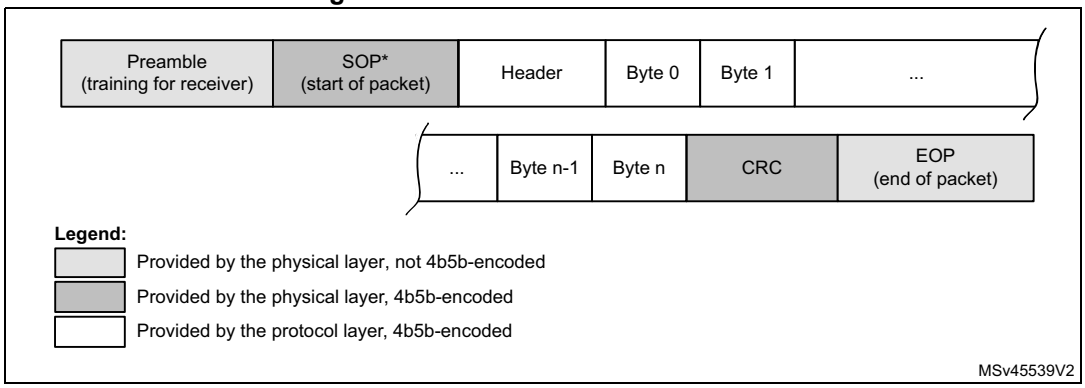


Packet format

Messages other than Hard Reset and Cable Reset

The packet format is shown in the following figure, with information on 4b5b encode and data source.

Figure 955. Packet format



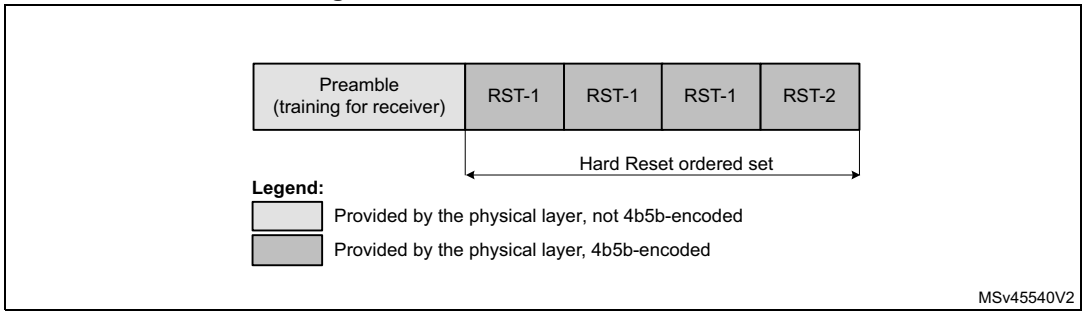
Hard Reset

The physical layer handles the Hard Reset signaling differently than the other types of message as it has higher priority to be able to interrupt an ongoing transfer.

The physical layer specification implies the following sequence in the case of an ongoing Tx message:

1. Terminate the message by sending an EOP K-code and discard the rest of the message.
2. Wait for *tInterFrameGap* time.
3. If the CC line is not idle, wait until it goes idle.
4. Send the preamble followed by the four K-codes of Hard Reset signaling.
5. Disable the CC channel (stop sending and receiving), reset the physical layer and inform the protocol layer that the physical layer is reset.
6. Re-enable the channel when requested by the protocol layer.

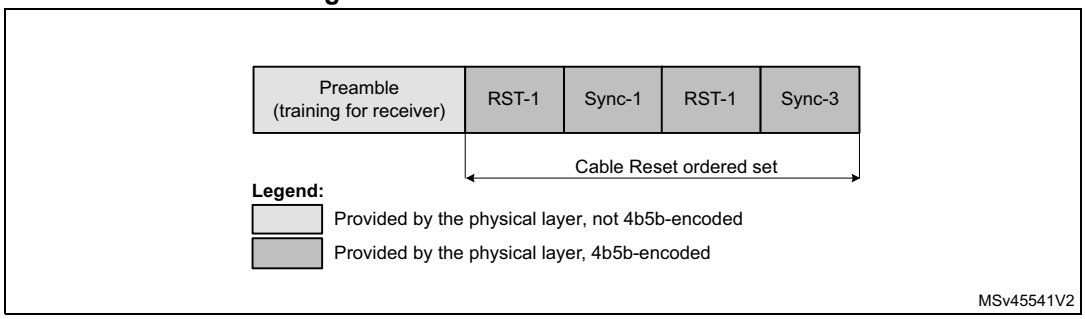
Figure 956. Line format of Hard Reset



Cable Reset

Cable Reset shown in the following figure is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high-priority treatment.

Figure 957. Line format of Cable Reset



Collision avoidance

The physical layer respects the *tInterFrameGap* delay between end of last-transmitted bit of a Tx message, and the first bit of a following message.

It also checks the idle state of the CC line before starting transmission. The CC line is considered idle if it shows less than three (*nTransitionCount*) transitions within *tTransitionWindow* (12 to 20 µs). The Power Delivery specification revision 3.1 also requires to manage the Rp value (source) and monitor Type-C voltage level for these Rp modifications (at the sink).

Physical layer signaling schemes

The bit are signaled with bi-phase mark coding (BMC).

BIST

Depending on the BIST action required by the protocol layer, either of the following can be run:

- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to “BMC” mode) are:

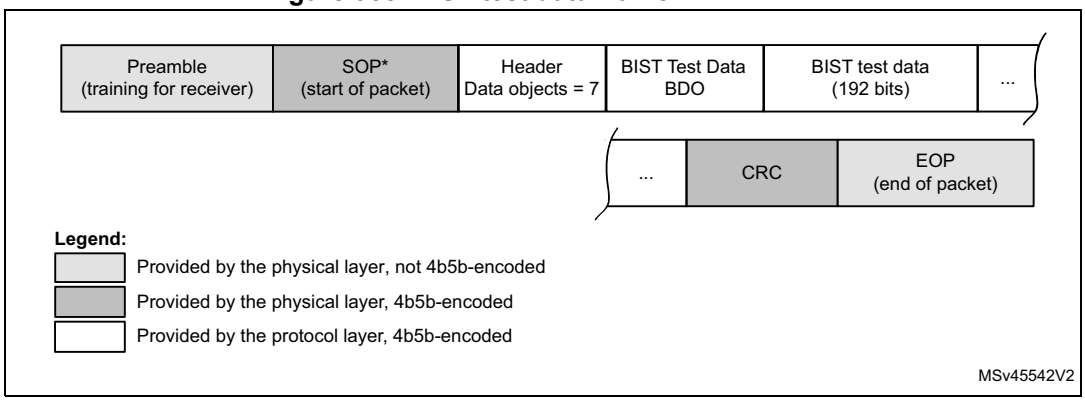
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only. As opposed to Tx, the receiver in this mode simply ignores the CC line during this state.

BIST test data pattern

The test data pattern is not viewed as a special case in UCPD.

The BIST test data packet frame format is shown in the following figure.

Figure 958. BIST test data frame



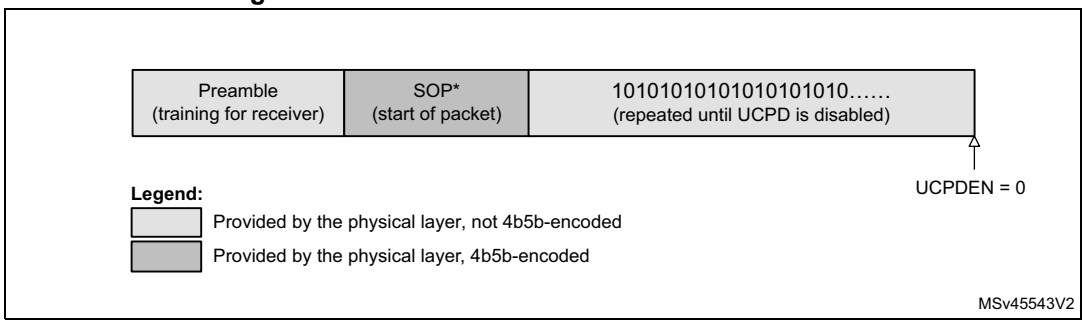
This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 955: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block's behavior) that differentiates the general packet from the BIST Test Data packet.

BIST Carrier Mode 2

When required, this BIST test mode sends an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification, the protocol layer defines a counter that indicates when to exit this mode.

The way to quit the infinite 1010 sequence (according to requirements of the USB PD specification) is to disable the UCPD peripheral via the UCPDEN bit.

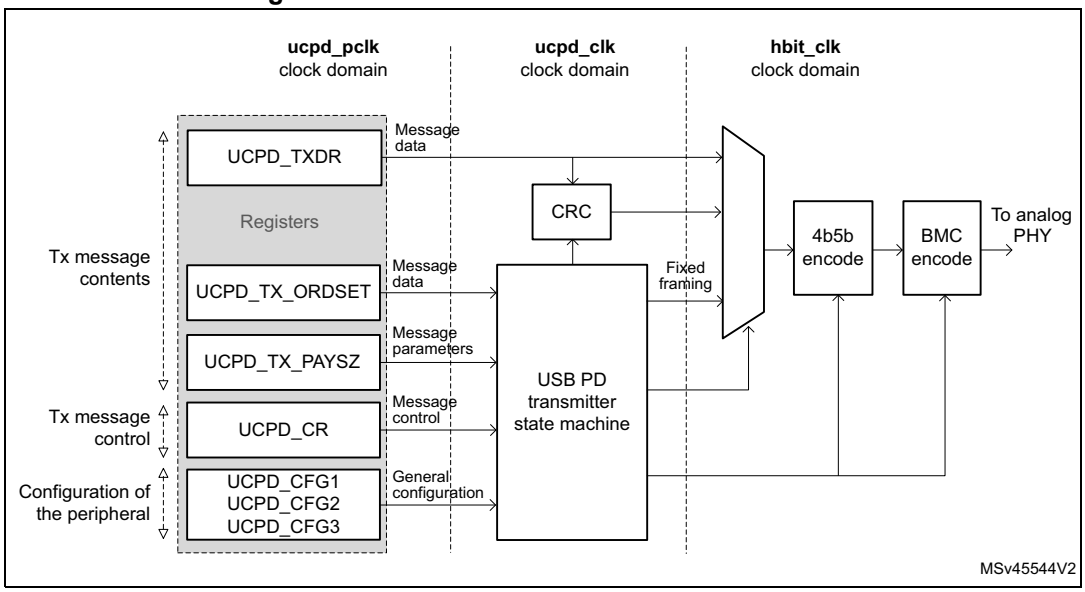
Figure 959. BIST Carrier Mode 2 frame



59.4.4 UCPD BMC transmitter

The BMC transmitter comprises 4b5b encoding, CRC generation, and BMC encode, as shown in the following figure. Its output goes to the analog PHY through a channel switch.

Figure 960. UCPD BMC transmitter architecture



BMC encoder

The bi-phase mark coding method is defined in the *IEC 60958-1 Digital Audio Interface Part:1 General Edition 3.0 2008-09 www.iec.ch* specification.

The half-bit clock hbit_clk is derived from ucpd_clk through a simple divider controlled by the HBITCLKDIV[5:0] bitfield of the UCPD_CFGR1 register. This ensures the same duration of high and low half-bit periods (if neglecting a small difference due to different rising and falling edge duration and due to jitter), and the same bit duration (if neglecting jitter).

Transmitter timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

– *tInterFrameGap*: via IFRGAP (pre-defined value, can be altered)
– *tTransitionWindow*: via TRANSWIN (pre-defined value, can be altered)

These two counters once set correctly generates the interframe gap.

Hard Reset in transmitter

In order to facilitate generation of a Hard Reset, a special code of TXMODE field is used. No other fields need to be written.

On writing the correct code, the hardware forces Hard Reset Tx under the correct (optimal) timings with respect to an ongoing Tx message, which (if still in progress) is cleanly terminated by truncating the current sequence and directly appending an EOP K-code sequence. No specific interrupt is generated relating to this truncation event.

Transmitter behavior in the case of errors

The under-run condition (TXUND interrupt) may happen by accident and in this case, the UCPD is starved of (the correct) Tx payload and is not able to complete the Tx message correctly. This is a serious error (for this to happen the software fails to respond in time). As a result the hardware ensures the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.

59.4.5 UCPD BMC receiver

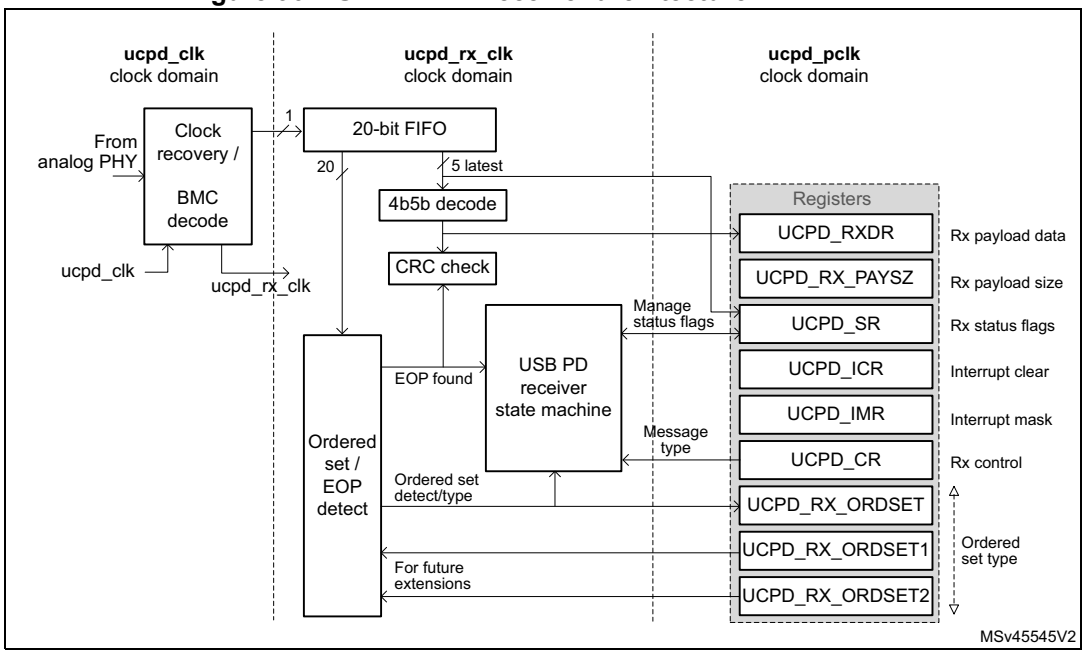
The UCPD BMC receiver performs:

- Clock recovery
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-code ordered set recognition
- CRC checking
- SOP detection
- EOP detection

The receiver is activated as soon as the UCPD peripheral is enabled (via UCPDEN), but it waits for an idle CC line state before attempting to receive a message.

The following figure shows the UCPD BMC receiver high-level architecture.

Figure 961. UCPD BMC receiver architecture



CRC checker

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR is set true and the receive data must be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR is generated in place of RXMSGEND.

Ordered set detection

This function detects the different ordered sets each consisting of four 5-bit K-codes.

Once we are in the preamble we opens a sliding window detection of the ordered set (4 words of 5 bits).

The ordered sets detected include all SOP* codes (SOP, SOP', and SOP''), but also Hard Reset, Cable Reset, SOP'_Debug, SOP''_Debug, and two extensions defined by registers UCPD_RX_ORDEXT1 and UCPD_RX_ORDEXT2.

EOP detection and Hard Reset exception handling

EOP is a fixed 5-bit K-code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (if a previous message transmit is still in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection can be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx message is arriving. When an "early EOP" is detected, the truncated Rx message is immediately discarded.

Truncated or corrupted message exception

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which is completed with a CRC and EOP. If at any point during these phases an error condition happens:

- the line becomes static for a time significantly longer than one "UI" period (the exact threshold for this condition is not critical but the exception must occur before three UIs), or
- the message goes to the end but it is not recognized (for example EOP is corrupted).

In both cases, the receiver quits the current message, raising RXMSGEND and RXERR flags.

Short preamble or incomplete ordered set exception

In the exceptional case of the receiver seeing less that half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine does not start.

In both of these cases, the clock-recovery/BMC decoder re-starts, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.

59.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)

UCPD offers simple control of these resistors via ANAMODE and ANASUBMODE[1:0]. In case only one of the CC lines is to be used, it is possible to optimize power consumption by disabling control on the other line, through the CCENABLE[1:0] bitfield.

When the MCU is unpowered, it still presents the "dead battery" Rd, provided that UCPDx_DBCC1 and UCPDx_DBCC2 pins are each connected to UCPDx_CC1 and UCPDx_CC2 pins, respectively.

If dead battery behavior is not required (for example for source only products), then UCPDx_DBCC1 and UCPDx_DBCC2 pins must both be tied to ground.

After power arrives and the MCU boots, the desired behavior (for example source) must be programmed into ANAMODE and ANASUBMODE[1:0] before setting the UCPD_DBDIS bit of the PWR_UCPDR register to remove dead battery pull-down resistor and allow the values just programmed to take effect.

Use of Standby low-power mode is possible for sinks in the unattached state.

59.4.7 UCPD Type-C voltage monitoring and de-bouncing

For correct operation of the Type-C state machine and for detecting the cable orientation, the CC1/2 lines must be monitored for voltage level, while ignoring fast events such as peaks.

Thresholds between voltage levels on the CC1/2 lines are determined through PHY threshold detector settings.

The TYPEC_VSTATE_CC1/2[1:0] bitfields reflect the CC1/2 line levels processed with a hardware de-bouncing filter that suppresses high-speed line events such as peaks. The PHYCCSEL bit selects the line, CC1 or CC2, to be used for Power Delivery signaling.

For minimizing the power consumption, it is recommended to use the polling method, with the Type-C detectors only turned on for the instant of polling, rather than keeping the Type-C detectors permanently on and wake the device up from Stop mode upon CC1/2 line events.

59.4.8 UCPD fast role swap (FRS)

FRS signaling

The FRS condition (a pulse of a specific length), is generated upon setting the FRSTX bit.

For the duration of FRS condition, the I/O configured as UCPD_FRSTX (alternate function) controls, with high level, the gate of an external NMOS transistor that pulls the active CC line down.

FRS detection

FRS monitoring is enabled by setting the bit FRSRXEN, after writing PHYCCSEL that selects the active CC line depending on the cable orientation detected.

59.4.9 UCPD DMA Interface

DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle UCPD1_TXDR and UCPD1_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

59.4.10 Wake-up from Stop mode

For power consumption optimization, it is useful to use Stop mode and wait for events on CC lines to wake the MCU up.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.

The events causing the wake-up can be:

- Events on the BMC receiver (RXORDDET, RXHRSTDET), hardware enable PHYRXEN
- Event on the FRS detector (FRSEVT), hardware enable FRSRXEN
- Events on the Type-C detectors (TYPECEVT1, TYPECEVT2), hardware enables CC1TCDIS, CC2TCDIS

59.5 UCPD programming sequences

The normal sequence of use of the UCPD unit is:

1. Configure UCPD.
2. Enable UCPD.
3. Concurrently:
   – On demand from protocol layer, send Tx message
   – Intercept (poll or wait for interrupt) relevant Rx messages and recover detail to hand off to protocol layer

Repeat the last point infinitely.

59.5.1 Initialization phase

Use the following sequence for a clean startup:

1. Prepare all initial clock divider values, by writing the UCPD_CFG register.
2. Enable the unit, by setting the UCPDEN bit.
3. Enable the analog Rx filter of either CC line, via the RXAFILTEN bit of the UCPD_CFGR2 register.

59.5.2 Type-C state machine handling

For the general application cases of source, sink, or dual-role port (the last alternating the source and the sink), the software must implement a corresponding USB Type-C state machine. The basic coding is in the following table.

Table 695. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx

ANAMODE	ANASUBMODE[1:0]	Notes	TYPEC_VSTATE_CCx[1:0]			
			00	01	10	11
0: Source	00: Disabled	Disabled	N/A			
	01: Default USB Rp	-	vRa[Def]	vRd[Def]	vOPEN[Def]	N/A
	10: 1.5A Rp	-	vRa[1.5]	vRd[1.5]	vOPEN[1.5]	
	11: 3.0A Rp	-	vRa[3.0]	vRd[3.0]	vOPEN[3.0]	
1: Sink	xx	-	vRa	vRd-USB	vRd-1.5	vRd-3.0

The CCENABLE[1:0] bitfield can disable pull-up/pull-downs on one of the CC lines.

Note: The Type-C state machine depends not only on CC line levels, but also on VBUS presence detection (sink mode) and, when in source mode, determines VCONN generation and

VBUS state (ON/OFF/+voltage level); discharge). UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be connected to the CC line), but the application needs these inputs and controls, to function correctly.

General programming sequence (with UCPD configured then enabled)

1. Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and defines the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming, the PHY is effectively off.
2. Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink).
3. In the case of no connection, wait for a connection event.
4. Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages.
5. When a new interrupt/event occurs on PHYEVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine.

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behavior from that time] PHYEVT1/2 occurs and the TYPEC_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):

Table 696. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)

Type-C state	ANAMODE; ANASUBMODE[1:0]	CCENABLE	PHYCCSEL	RDCH	CC[x] VCONNEN[(1)]	Event => go to next line	Comments
Unattached. SRC	0:Source; 11:Rp3A0	11:both enabled	0 (don't care)	0: [Normal]	00: [neither]	PHYEVT 1: [VRd-3A0]	Wait for sink attach detect ; seen on CC1 [EVT1]
Attachwait. SRC			0 [Rd on CC1]			PHYEVT 2: [VRa]	Attachwait started (100-200 ms) ; now also see the Ra => requesting VCONN
Attached. SRC [VCONN => CC2]	0:Source; 11:Rp3A0 [SinkTxOK]	01: CC2 disable (possible and recommended due to external VCONN switch)			10: [CC2 active]	Timer (100 ms) and no PHYEVTx	Local CC2 disconnected from PHY (VCONN switch connects VCONN source to CC2 externally; Continue to monitor PHYEVT1
	0:Source; 10:Rp1A5 [SinkTxNG]					SW timers (SinkTxNG)	Source wants to initiate message sequence (SinkTxNG condition set first)
	0:Source; 11:Rp3A0 [SinkTxOK]						Source finished message sequence (SinkTxOK condition afterwards)
						PHYEVT 1: [VOpen-3A0]	Wait for Sink disconnected (Vopen on CC1)
Unattached wait. SRC		11:both enabled		1: [discharge]	00: [neither]	> 0.8V detection	Discharge VCONN [CC2] actively [Rdch]; to < 0.8V
Unattached. SRC	0:Source; 11:Rp3A0			0: [Normal]			[Details as first line of table]

1. Two GPIOs to enable VCONN through external load switch components

59.5.3 USB PD transmit

On reception of a message from the protocol layer (that is, to be sent), prepare Tx message contents by writing the UCPD_TX_ORDSET and UCPD_TX_PAYSZ registers.

The message transmission is triggered by setting the TXSEND bit, with an appropriate value of the TXMODE bitfield.

When the data byte is transmitted, the TXIS flag is raised to request a new data write to the UCPD_TXDR register.

This re-iterates until the entire payload of data is transmitted.

Upon sending the CRC packet, the TXMSGSENT flag is set to indicate the completion of the message transmission.

Hard Reset transmission

As soon as it is known that a Hard Reset needs to be transmitted, setting the TXHRST bit of the UCPD_CR register forces the internal state machine to generate the correct sequence. The value of UCPD_TX_ORDSET does not require update in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case, for example, UCPD truncates the payload of the current message, appending EOP to the end. No notification is available via the registers (for example through the TXMSGSEND flag). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it is completed).

Use of DMA for transmission

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD_CR register.

For each message:

- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus a number of data bytes corresponding to the number of data words multiplied by four
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (that is, when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

59.5.4 USB PD receive

Notification of start of the receive message sequence is triggered by an interrupt on UCPD_SR (bit RXORDDET).

The information is recovered by reading:

- UCPD_RX_SOP (on interrupt RXORDDET)
- UCPD_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD_RXPAYSZ (on interrupt RXMSGEND)

The data previously read from UCPD_RXDR above must be discarded at this point if the RXERR flag is set.

If the CRC is valid, the received data is transferred to the protocol layer.

For debug purposes, it may be desirable to track statistics of the number of incorrect K-codes received (this is done only when 3/4 K-codes were valid as defined in the specification). This is facilitated through:

- RXSOP3OF4 bit indicating the presence of at least one invalid K-code
- RXSOPKINVALID bitfield identifying the order of invalid K-code in the ordered set

Use of DMA for reception

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD_CR register.

Whenever a Rx message is expected:

- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support).
- After receiving RXORDDET, DMA operation starts working in the background.
- On reception of RXMSGEND interrupt, read RXPAYSZ.
- Double-check RXPAYSZ vs. the number of DMA Rx bytes (must correspond but DMA read of RXDR is slightly after RXDR gets last byte).
- Process the DMA Rx buffer.
- Prepare next Rx DMA buffer as soon as possible in order to be ready.

59.5.5 UCPD software trimming

The CC pull-up (Rp) and pull-down (Rd) devices must be trimmed on each part, to meet the required accuracy. The trimming values are saved in the nonvolatile memory.

To trim the CC pull-up and pull-down devices by software, apply the following procedure:

1. Retrieve the trim values from the nonvolatile memory (refer to *Table 688: UCPD software trim data*)
2. At initialization, write the trim values to the bitfields of the *UCPD configuration register 3 (UCPD_CFGR3)* as follows:
   - 3A0_CC1[3:0] to TRIM_CC1_RP[3:0]
   - 3A0_CC2[3:0] to TRIM_CC2_RP[3:0]
   - 1A5_CC1[3:0] to TRIM_CC1_RP1A5[3:0]
   - 1A5_CC2[3:0] to TRIM_CC2_RP1A5[3:0]
   - Rd_CC1[3:0] to TRIM_CC1_RD[3:0]
   - Rd_CC2[3:0] to TRIM_CC2_RD[3:0]

59.6 UCPD low-power modes

A summary of low-power modes is shown below in *Table 697: Effect of low power modes on the UCPD*.

Table 697. Effect of low power modes on the UCPD

Mode	Description
Sleep	No effect
Stop	Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.
Standby	UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.
Unpowered	Dead battery pull-downs remain active.

The UCPD is able to wake up the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wake-up from Stop mode is enabled by setting the WUPEN bit in the UCPD_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which can later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which can potentially generate an Rx message event (that is, RXORDSET) later
- Activity on Power Delivery FRS detector which can potentially generate an FRS signaling detection event (that is, FRSEVT) later

It order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)
- FRS detection signal (from the selected CC line)

59.7 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

Table 698. UCPD interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
FRS detection	FRSEVT	Set FRSEVTCF	FRSEVTIE
Type C voltage level change on CC2	TYPECEVT2	Set TYPECEVT2CF	TYPECEVT2IE
Type C voltage level change on CC1	TYPECEVT1	Set TYPECEVT1CF	TYPECEVT1IE
Rx message received	RXMSGEND	Set RXMSGENDCF	RXMSGENDIE
Rx data overflow	RXOVR	Set RXOVRCF	RXOVR
Rx Hard Reset detected	RXHRSTDET	Set RXHRSTDETCF	RXHRSTDETIE
Rx ordered set (4 K-codes) detected	RXORDDET	Set RXORDDETCF	RXORDDETIE
Receive data register not empty	RXNE	Read data in UCPD_RXDR	RXNEIE
Tx data underrun	TXUND	Set TXUNDCF	TXUNDIE
Hard Reset sent	HRSTSENT	Set HRSTSENTCF	HRSTSENTIE
Hard Reset discarded	HRSTDISC	Set HRSTDISCCF	HRSTDISCIE
Transmit message aborted	TXMSGABT	Set TXMSGABTCF	TXMSGABTIE

Table 698. UCPD interrupt requests (continued)

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Transmit message sent	TXMSGSENT	Set TXMSGSENTCF	TXMSGSENTIE
Transmit message discarded	TXMSGDISC	Set TXMSGDISCCF	TXMSGDISCIE
Transmit data required	TXIS	Write data to the UCPD_TXDR register	TXISIE

When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD_SR register.

Depending on which bit is at 1, the ISR must handle that condition and clear the bit by a write to the appropriate bit of the UCPD_ICR register.

59.8 UCPD registers

59.8.1 UCPD configuration register 1 (UCPD_CFGR1)

Address offset: 0x000

Reset value: 0x0000 0000

General configuration of the peripheral. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_UCPDCLK[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UCPDEN**: UCPD peripheral enable

General enable of the UCPD peripheral.

0: Disable

1: Enable

Upon disabling, the peripheral instantly quits any ongoing activity and all control bits and bitfields default to their reset values. They must be set to their desired values each time the peripheral transits from disabled to enabled state.

Bit 30 **RXDMAEN**: Reception DMA mode enable

When set, the bit enables DMA mode for reception.

0: Disable

1: Enable

Bit 29 **TXDMAEN**: Transmission DMA mode enable

When set, the bit enables DMA mode for transmission.

0: Disable

1: Enable

Bits 28:20 **RXORDSETEN[8:0]**: Receiver ordered set enable

The bitfield determines the types of ordered sets that the receiver must detect. When set/cleared, each bit enables/disables a specific function:
0bXXXXXXXX1: SOP detect enabled
0bXXXXXXX1X: SOP' detect enabled
0bXXXXXX1XX: SOP'' detect enabled
0bXXXXX1XXX: Hard Reset detect enabled
0bXXXX1XXXX: Cable Detect reset enabled
0bXXX1XXXXX: SOP'_Debug enabled
0bXX1XXXXXX: SOP''_Debug enabled
0bX1XXXXXXX: SOP extension#1 enabled
0b1XXXXXXXX: SOP extension#2 enabled

Bits 19:17 **PSC_UCPDCLK[2:0]**: Pre-scaler division ratio for generating ucpd_clk

The bitfield determines the division ratio of a kernel clock pre-scaler producing UCPD peripheral clock (ucpd_clk).
0x0: 1 (bypass)
0x1: 2
0x2: 4
0x3: 8
0x4: 16
It is recommended to use the pre-scaler so as to set the ucpd_clk frequency in the range from 6 to 12 MHz.

Bit 16 Reserved, must be kept at reset value.

Bits 15:11 **TRANSWIN[4:0]**: Transition window duration

The bitfield determines the division ratio (the bitfield value plus one) of a hbit_clk divider producing *tTransitionWindow* interval.
0x00: Not supported
0x01: 2
0x09: 10 (recommended)
0x1F: 32
Set a value that produces an interval of 12 to 20 us, taking into account the ucpd_clk frequency and the HBITCLKDIV[5:0] bitfield setting.

Bits 10:6 **IFRGAP[4:0]**: Division ratio for producing inter-frame gap timer clock

The bitfield determines the division ratio (the bitfield value plus one) of a ucpd_clk divider producing inter-frame gap timer clock (*tInterFrameGap*).
0x00: Not supported
0x01: 2
0x0D: 14
0x0E: 15
0x0F: 16
0x1F: 32
The division ratio 15 is to apply for Tx clock at the USB PD 2.0 specification nominal value. The division ratios below 15 are to apply for Tx clock below nominal, and the division ratios above 15 for Tx clock above nominal.

Bits 5:0 **HBITCLKDIV[5:0]**: Division ratio for producing half-bit clock

The bitfield determines the division ratio (the bitfield value plus one) of a ucpd_clk divider producing half-bit clock (hbit_clk).
0x00: 1 (bypass)
0x0C: 13
0x1A: 27
0x3F: 64

59.8.2 UCPD configuration register 2 (UCPD_CFGR2)

Address offset: 0x004

Reset value: 0x0000 0000

Configuration of the UCPD Rx signal filtering. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXAFILTEN	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
							rw					rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RXAFILTEN**: Rx analog filter enable
Setting the bit enables the Rx analog filter required for optimum Power Delivery reception.
0: Disable
1: Enable

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **WUPEN**: Wake-up from Stop mode enable
Setting the bit enables the UCPD_ASYNC_INT signal.
0: Disable
1: Enable

Bit 2 **FORCECLK**: Force ClkReq clock request
0: Do not force clock request
1: Force clock request

Bit 1 **RXFILT2N3**: BMC decoder Rx pre-filter sampling method
Number of consistent consecutive samples before confirming a new value.
0: 3 samples
1: 2 samples

Bit 0 **RXFILTDIS**: BMC decoder Rx pre-filter enable
0: Enable
1: Disable
The sampling clock is that of the receiver (that is, after pre-scaler).

59.8.3 UCPD configuration register 3 (UCPD_CFGR3)

Address offset: 0x008

Reset value: 0x0000 0000

Configuration of UCPD trimming of the CC pull-up and pull-down devices. The trimming is managed by hardware until the first software write into this register.

The register is reserved (must not be written) for devices that support the fully automatic trimming. Refer to *Section 59.3: UCPD implementation*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TRIM_CC2_RP[3:0]				Res.	TRIM_CC2_RP1A5[3:0]				TRIM_CC2_RD[3:0]			
			rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIM_CC1_RP[3:0]				Res.	TRIM_CC1_RP1A5[3:0]				TRIM_CC1_RD[3:0]			
			rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **TRIM_CC2_RP[3:0]**: SW trim value for Rp current sources on the CC2 line

Bit 24 Reserved, must be kept at reset value.

Bits 23:20 **TRIM_CC2_RP1A5[3:0]**: SW trim value for Rp current sources on the CC2 line (best trim for RP1A5 mode)

Bits 19:16 **TRIM_CC2_RD[3:0]**: SW trim value for Rd resistor on the CC2 line

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:9 **TRIM_CC1_RP[3:0]**: SW trim value for Rp current sources on the CC1 line

Bit 8 Reserved, must be kept at reset value.

Bits 7:4 **TRIM_CC1_RP1A5[3:0]**: SW trim value for Rp current sources on the CC2 line (best trim for RP1A5 mode)

Bits 3:0 **TRIM_CC1_RD[3:0]**: SW trim value for Rd resistor on the CC1 line

59.8.4 UCPD control register (UCPD_CR)

Address offset: 0x00C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN
										rw	rw		rw	rs	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
				rw	rw	rw	rw	rw	rw	rw	rw	rs	rs	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC2TCDIS**: CC2 Type-C detector disable

The bit disables the Type-C detector on the CC2 line.

0: Enable

1: Disable

When enabled, the Type-C detector for CC2 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 20 **CC1TCDIS**: CC1 Type-C detector disable
The bit disables the Type-C detector on the CC1 line.
0: Enable
1: Disable
When enabled, the Type-C detector for CC1 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 19 Reserved, must be kept at reset value.

Bit 18 **RDCH**: Rdch condition drive
The bit drives Rdch condition on the CC line selected through the PHYCCSEL bit (thus associated with VCONN), by remaining set during the source-only *UnattachedWait.SRC* state, to respect the Type-C state. Refer to *"USB Type-C ECN for Source VCONN Discharge"*. The CCENABLE[1:0] bitfield must be set accordingly, too.
0: No effect
1: Rdch condition drive

Bit 17 **FRSTX**: FRS Tx signaling enable.
Setting the bit enables FRS Tx signaling.
0: No effect
1: Enable
The bit is cleared by hardware after a delay respecting the USB Power Delivery specification Revision 3.1.

Bit 16 **FRSRXEN**: FRS event detection enable
Setting the bit enables FRS Rx event (FRSEVT) detection on the CC line selected through the PHYCCSEL bit. 0: Disable
1: Enable
Clear the bit when the device is attached to an FRS-incapable source/sink.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 **CCENABLE[1:0]**: CC line enable
This bitfield enables CC1 and CC2 line analog PHYs (pull-ups and pull-downs) according to ANAMODE and ANASUBMODE[1:0] setting.
0x0: Disable both PHYs
0x1: Enable CC1 PHY
0x2: Enable CC2 PHY
0x3: Enable CC1 and CC2 PHY
A single line PHY can be enabled when, for example, the other line is driven by VCONN via an external VCONN switch. Enabling both PHYs is the normal usage for sink/source.

Bit 9 **ANAMODE**: Analog PHY operating mode
0: Source
1: Sink
The use of CC1 and CC2 depends on CCENABLE. Refer to *Table 695: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield in conjunction with ANASUBMODE[1:0].

Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode
Refer to *Table 695: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield.

Bit 6 **PHYCCSEL**: CC1/CC2 line selector for USB Power Delivery signaling

0: Use CC1 IO for Power Delivery communication

1: Use CC2 IO for Power Delivery communication

The selection depends on the cable orientation as discovered at attach.

Bit 5 **PHYRXEN**: USB Power Delivery receiver enable

0: Disable

1: Enable

Both CC1 and CC2 receivers are disabled when the bit is cleared. Only the CC receiver selected via the PHYCCSEL bit is enabled when the bit is set.

Bit 4 **RXMODE**: Receiver mode

Determines the mode of the receiver.

0: Normal receive mode

1: BIST receive mode (BIST test data mode)

When the bit is set, RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as for a normal message. As this mode prevents reception of the header (containing MessageID), software has to auto-increment a received MessageID counter for inclusion in the GoodCRC acknowledge that must still be transmitted during this test.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset

0: No effect

1: Start Tx Hard Reset message

The bit is cleared by hardware as soon as the message transmission begins or is discarded.

Bit 2 **TXSEND**: Command to send a Tx packet

0: No effect

1: Start Tx packet transmission

The bit is cleared by hardware as soon as the packet transmission begins or is discarded.

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet

Writing the bitfield triggers the action as follows, depending on the value:

0x0: Transmission of Tx packet previously defined in other registers

0x1: Cable Reset sequence

0x2: BIST test sequence (BIST Carrier Mode 2)

Others: invalid

From V1.1 of the USB PD specification, there is a counter defined for the duration of the BIST Carrier Mode 2. To quit this mode correctly (after the "tBISTContMode" delay), disable the peripheral (UCPDEN = 0).

59.8.5 UCPD interrupt mask register (UCPD_IMR)

Address offset: 0x010

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTIE	Res.	Res.	Res.	Res.
											r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTIE**: FRSEVT interrupt enable
0: Disable
1: Enable

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2IE**: TYPECEVT2 interrupt enable
0: Disable
1: Enable

Bit 14 **TYPECEVT1IE**: TYPECEVT1 interrupt enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDIE**: RXMSGEND interrupt enable
0: Disable
1: Enable

Bit 11 **RXOVRIE**: RXOVR interrupt enable
0: Disable
1: Enable

Bit 10 **RXHRSTDETIE**: RXHRSTDET interrupt enable
0: Disable
1: Enable

Bit 9 **RXORDDETIE**: RXORDDET interrupt enable
0: Disable
1: Enable

Bit 8 **RXNEIE**: RXNE interrupt enable
0: Disable
1: Enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUNDIE**: TXUND interrupt enable
0: Disable
1: Enable

Bit 5 **HRSTSENTIE**: HRSTSENT interrupt enable
0: Disable
1: Enable

Bit 4 **HRSTDISCIE**: HRSTDISC interrupt enable
0: Disable
1: Enable

Bit 3 **TXMSGABTIE**: TXMSGABT interrupt enable
0: Disable
1: Enable

Bit 2 **TXMSGSENTIE**: TXMSGSENT interrupt enable
0: Disable
1: Enable

Bit 1 **TXMSGDISCIE**: TXMSGDISC interrupt enable
0: Disable
1: Enable

Bit 0 **TXISIE**: TXIS interrupt enable
0: Disable
1: Enable

59.8.6 UCPD status register (UCPD_SR)

Address offset: 0x014

Reset value: 0x0000 0000

The flags (single-bit status bitfields) are associated with interrupt request. Interrupt is generated if enabled by the corresponding bit of the UCPD_IMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]	
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
r	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVT**: FRS detection event
The flag is cleared by setting the FRSEVTCF bit.
0: No new event
1: New FRS receive event occurred

Bits 19:18 **TYPEC_VSTATE_CC2[1:0]**: CC2 line voltage level
The status bitfield indicates the voltage level on the CC2 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC2 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bits 17:16 **TYPEC_VSTATE_CC1[1:0]**:
The status bitfield indicates the voltage level on the CC1 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC1 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bit 15 **TYPECEVT2**: Type-C voltage level event on CC2 line
The flag indicates a change of the TYPEC_VSTATE_CC2[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 14 **TYPECEVT1**: Type-C voltage level event on CC1 line
The flag indicates a change of the TYPEC_VSTATE_CC1[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 13 **RXERR**: Receive message error
The flag indicates errors of the last Rx message declared (via RXMSGEND), such as incorrect CRC or truncated message (a line becoming static before EOP is met). It is asserted whenever the RXMSGEND flag is set.
0: No error detected
1: Error(s) detected

Bit 12 **RXMSGEND**: Rx message received
The flag indicates whether a message (except Hard Reset message) has been received, regardless the CRC value. The flag is cleared by setting the RXMSGENDCF bit.
0: No new Rx message received
1: A new Rx message received
The RXERR flag set when the RXMSGEND flag goes high indicates errors in the last-received message.

Bit 11 **RXOVR**: Rx data overflow detection
The flag indicates Rx data buffer overflow. It is cleared by setting the RXOVRCF bit.
0: No overflow
1: Overflow
The buffer overflow can occur if the received data are not read fast enough.

Bit 10 **RXHRSTDET**: Rx Hard Reset receipt detection
The flag indicates the receipt of valid Hard Reset message. It is cleared by setting the RXHRSTDETCF bit.
0: Hard Reset not received
1: Hard Reset received

Bit 9 **RXORDDET**: Rx ordered set (4 K-codes) detection
The flag indicates the detection of an ordered set. The relevant information is stored in the RXORDSET[2:0] bitfield of the UCPD_RX_ORDSET register. It is cleared by setting the RXORDDETCF bit.
0: No ordered set detected
1: A new ordered set detected

Bit 8 **RXNE**: Receive data register not empty detection

The flag indicates that the UCPD_RXDR register is not empty. It is automatically cleared upon reading UCPD_RXDR.

0: Rx data register empty

1: Rx data register not empty

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUND**: Tx data underrun detection

The flag indicates that the Tx data register (UCPD_TXDR) was not written in time for a transmit message to execute normally. It is cleared by setting the TXUNDCF bit.

0: No Tx data underrun detected

1: Tx data underrun detected

Bit 5 **HRSTSENT**: Hard Reset message sent

The flag indicates that the Hard Reset message is sent. The flag is cleared by setting the HRSTSENTCF bit.

0: No Hard Reset message sent

1: Hard Reset message sent

Bit 4 **HRSTDISC**: Hard Reset discarded

The flag indicates that the Hard Reset message is discarded. The flag is cleared by setting the HRSTDISCCF bit.

0: No Hard Reset discarded

1: Hard Reset discarded

Bit 3 **TXMSGABT**: Transmit message abort

The flag indicates that a Tx message is aborted due to a subsequent Hard Reset message send request taking priority during transmit. It is cleared by setting the TXMSGABTCF bit.

0: No transmit message abort

1: Transmit message abort

Bit 2 **TXMSGSENT**: Message transmission completed

The flag indicates the completion of packet transmission. It is cleared by setting the TXMSGSENTCF bit.

0: No Tx message completed

1: Tx message completed

In the event of a message transmission interrupted by a Hard Reset, the flag is not raised.

Bit 1 **TXMSGDISC**: Message transmission discarded

The flag indicates that a message transmission was dropped. The flag is cleared by setting the TXMSGDISCCF bit.

0: No Tx message discarded

1: Tx message discarded

Transmission of a message can be dropped if there is a concurrent receive in progress or at excessive noise on the line. After a Tx message is discarded, the flag is only raised when the CC line becomes idle.

Bit 0 **TXIS**: Transmit interrupt status

The flag indicates that the UCPD_TXDR register is empty and new data write is required (as the amount of data sent has not reached the payload size defined in the TXPAYSZ bitfield). The flag is cleared with the data write into the UCPD_TXDR register.

0: New Tx data write not required

1: New Tx data write required

59.8.7 UCPD interrupt clear register (UCPD_ICR)

Address offset: 0x018

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTCF	Res.	Res.	Res.	Res.
											w				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
w	w		w	w	w	w			w	w	w	w	w	w	

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTCF**: FRS event flag (FRSEVT) clear
Setting the bit clears the FRSEVT flag in the UCPD_SR register.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2CF**: Type-C CC2 line event flag (TYPECEVT2) clear
Setting the bit clears the TYPECEVT2 flag in the UCPD_SR register

Bit 14 **TYPECEVT1CF**: Type-C CC1 event flag (TYPECEVT1) clear
Setting the bit clears the TYPECEVT1 flag in the UCPD_SR register

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDCF**: Rx message received flag (RXMSGEND) clear
Setting the bit clears the RXMSGEND flag in the UCPD_SR register.

Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear
Setting the bit clears the RXOVR flag in the UCPD_SR register.

Bit 10 **RXHRSTDETCF**: Rx Hard Reset detect flag (RXHRSTDET) clear
Setting the bit clears the RXHRSTDET flag in the UCPD_SR register.

Bit 9 **RXORDDETCF**: Rx ordered set detect flag (RXORDDET) clear
Setting the bit clears the RXORDDET flag in the UCPD_SR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear
Setting the bit clears the TXUND flag in the UCPD_SR register.

Bit 5 **HRSTSENTCF**: Hard reset send flag (HRSTSENT) clear
Setting the bit clears the HRSTSENT flag in the UCPD_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discard flag (HRSTDISC) clear
Setting the bit clears the HRSTDISC flag in the UCPD_SR register.

Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear
Setting the bit clears the TXMSGABT flag in the UCPD_SR register.

Bit 2 **TXMSGSENTCF**: Tx message send flag (TXMSGSENT) clear
Setting the bit clears the TXMSGSENT flag in the UCPD_SR register.

Bit 1 **TXMSGDISCCF**: Tx message discard flag (TXMSGDISC) clear
Setting the bit clears the TXMSGDISC flag in the UCPD_SR register.

Bit 0 Reserved, must be kept at reset value.

59.8.8 UCPD Tx ordered set type register (UCPD_TX_ORDSETR)

Address offset: 0x01C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1) and no packet transmission is in progress (TXSEND and TXHRST bits are both low).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXORDSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **TXORDSET[19:0]**: Ordered set to transmit
The bitfield determines a full 20-bit sequence to transmit, consisting of four K-codes, each of five bits, defining the packet to transmit. The bit 0 (bit 0 of K-code1) is the first, the bit 19 (bit 4 of K-code4) the last.

59.8.9 UCPD Tx payload size register (UCPD_TX_PAYSZR)

Address offset: 0x020

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TXPAYSZ[9:0]**: Payload size yet to transmit

The bitfield is modified by software and by hardware. It contains the number of bytes of a payload (including header but excluding CRC) yet to transmit: each time a data byte is written into the UCPD_TXDR register, the bitfield value decrements and the TXIS bit is set, except when the bitfield value reaches zero. The enumerated values are standard payload sizes before the start of transmission.

0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)

59.8.10 UCPD Tx data register (UCPD_TXDR)

Address offset: 0x024

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]**: Data byte to transmit

59.8.11 UCPD Rx ordered set register (UCPD_RX_ORDSETR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP 3OF4	RXORDSET[2:0]		
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **RXSOPKINVALID[2:0]**:

The bitfield is for debug purposes only.

0x0: No K-code corrupted
0x1: First K-code corrupted
0x2: Second K-code corrupted
0x3: Third K-code corrupted
0x4: Fourth K-code corrupted
Others: Invalid

Bit 3 **RXSOP3OF4**:

The bit indicates the number of correct K-codes. For debug purposes only.

0: 4 correct K-codes out of 4
1: 3 correct K-codes out of 4

Bits 2:0 **RXORDSET[2:0]**: Rx ordered set code detected

0x0: SOP code detected in receiver
0x1: SOP' code detected in receiver
0x2: SOP'' code detected in receiver
0x3: SOP'_Debug detected in receiver
0x4: SOP''_Debug detected in receiver
0x5: Cable Reset detected in receiver
0x6: SOP extension#1 detected in receiver
0x7: SOP extension#2 detected in receiver

59.8.12 UCPD Rx payload size register (UCPD_RX_PAYSZR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **RXPAYSZ[9:0]**: Rx payload size received

This bitfield contains the number of bytes of a payload (including header but excluding CRC) received: each time a new data byte is received in the UCPD_RXDR register, the bitfield value increments and the RXMSGEND flag is set (and an interrupt generated if enabled).

0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)

The bitfield may return a spurious value when a byte reception is ongoing (the RXMSGEND flag is low).

59.8.13 UCPD receive data register (UCPD_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]**: Data byte received

59.8.14 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1)

Address offset: 0x034

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX1[19:0]**: Ordered set 1 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

59.8.15 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2)

Address offset: 0x038

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX2[19:0]**: Ordered set 2 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

59.8.16 UCPD register map

Table 699. UCPD register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	UCPD_CFGR1	UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_UCPDCLK[2:0]			Res.	TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	UCPD_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXAFILTEN	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
	Reset value																								0					0	0	0	0
0x008	UCPD_CFGR3	Res.	Res.	Res.	TRIM_CC2_RP[3:0]				Res.	TRIM_CC2_RP1A5[3:0]				TRIM_CC2_RD[3:0]				Res.	Res.	Res.	TRIM_CC1_RP[3:0]				Res.	TRIM_CC1_RP1A5[3:0]				TRIM_CC1_RD[3:0]			
	Reset value				0	0	0	0		0	0	0	0	0	0	0	0				0	0	0	0		0	0	0	0	0	0	0	0	0

Table 699. UCPD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	UCPD_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN	Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
	Reset value											0	0		0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x010	UCPD_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTIE	Res.	Res.	Res.	Res.	TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
	Reset value												0					0	0		0	0	0	0	0		0	0	0	0	0	0	0
0x014	UCPD_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]		TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x018	UCPD_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTCF	Res.	Res.	Res.	Res.	TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
	Reset value												0					0	0		0	0	0	0			0	0	0	0	0	0	
0x01C	UCPD_TX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	UCPD_TX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	UCPD_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x028	UCPD_RX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP3OF4	RXORDSET[2:0]		
	Reset value																										0	0	0	0	0	0	0
0x02C	UCPD_RX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0

Table 699. UCPD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	UCPD_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x034	UCPD_RX_ORDEXTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	UCPD_RX_ORDEXTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C - 0x3FF	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

60 Debug support (DBG)

60.1 Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/serial-wire debug access port, using industry-standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® CoreSight™ components.

- SWJ-DP: JTAG/serial-wire debug port
- AHB-AP: AHB access port
- ROM table
- System control space (SCS)
- Breakpoint unit (BPU)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded Trace Macrocell™ (ETM)
- Cross trigger interface (CTI)
- Trace port interface unit (TPU)

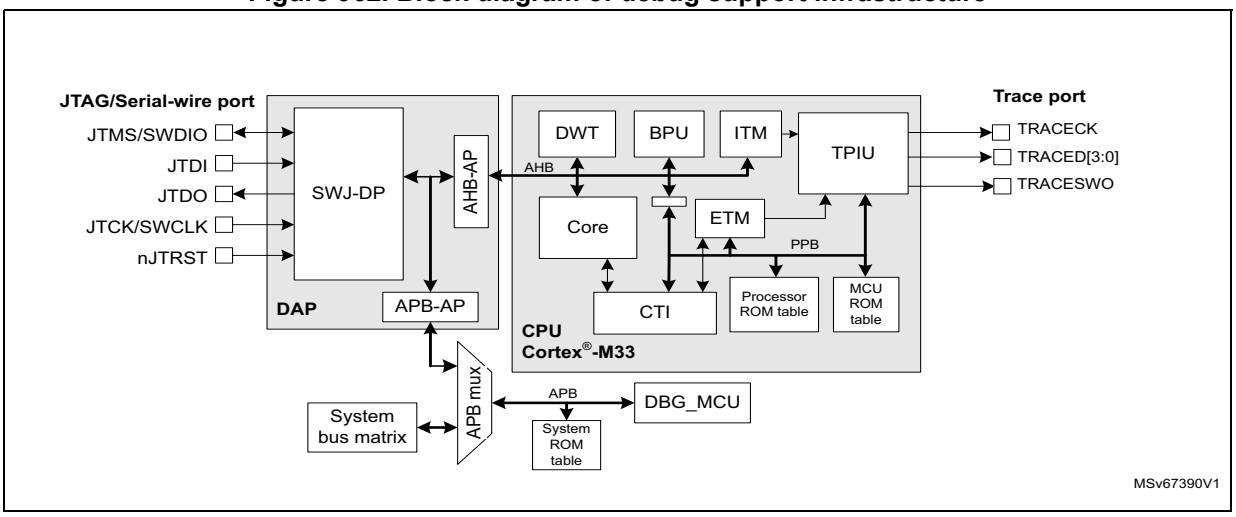
The debug features are accessible by the debugger via the AHB-AP.

Additional information can be found in the Arm® documents referenced in *Section 60.13*.

60.2 DBG functional description

60.2.1 DBG block diagram

Figure 962. Block diagram of debug support infrastructure



60.2.2 DBG pins and internal signals

Table 700. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial-wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial-wire clock	PA14
JTDI	I	JTAG test data input	-	-	PA15
JTDO	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

Table 701. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to the datasheet
TRACED1		Trace synchronous data out 1	
TRACED2		Trace synchronous data out 2	
TRACED3		Trace synchronous data out 3	
TRACECK		Trace clock	

Table 702. Single-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single-wire trace asynchronous data out	PB3(1)

1. TRACESWO is multiplexed with JTDO. This means that a single-wire trace is only available when using the serial-wire debug interface, and not when using JTAG.

60.2.3 DBG reset and clocks

The debug port (SWJ-DP) is reset by a power-on reset and when waking up from Standby mode.

The debugger supplies the clock for the debug port via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port. This clock must therefore continue to toggle for several cycles after access ends, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DCLK domain that covers the rest of the SWJ-DP and the access port.

The DCLK is a gated version of the system clock.

The DCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the *DP control and status register (DP_CTRLSTATR)*. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in DP_CTRL/STATR. The DCLK is disabled at power-up, and must be disabled when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor are clocked with the processor clock.

60.2.4 DBG power domains

The debug components are located in the core power domain. This means that the debugger connection is not possible in Shutdown or Standby low-power mode. To avoid losing the connection when the device enters Standby mode, the power can be maintained to the core by setting a bit in the *DBGMCU configuration register (DBGMCU_CR)*. This also keeps the processor clocks active and holds off the reset, so that the debug session is maintained.

60.2.5 Debug and low-power modes

The devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off or if the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power-saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power-saving mode, but the debugger does not lose the connection.

The emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information, refer to *Section 60.12*.

60.2.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. To protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope. For example, secure software debug and trace can be disabled without preventing the debug of nonsecure code.

Debugger access to secure memory (when permitted) must be performed using secure transactions on the debug AHB, that is, with the PROT[6] bit set in the *AP1 control/status word register (AP1_CSWR)*.

Debugger access is disabled while the processor is booting from system flash memory.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features

  0: All debug features are disabled.

  1: Debug features in nonsecure state are enabled. Debug features in secure state are dependent on the state of the **spiden** signal.
- **spiden**: enables debug in secure state when **dbgen** = 1.

  0: Debug features are disabled in secure state.

  1: Debug features are enabled in secure state.
- **niden**: enables trace and performance monitoring (noninvasive debug).

  0: Trace generation is disabled.

  1: Trace generation in nonsecure state is enabled. Trace generation in secure state is dependent on the state of the **spniden** signal.
- **spniden**: enables trace and performance monitoring in secure state when **niden** = 1.

  0: Trace generation is disabled in secure state.

  1: Trace generation is enabled in secure state.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant chapter, or to the relevant Arm® technical documentation.

The state of the signals are set according to the debug state as shown in *Table 703*, when TrustZone® is enabled (TZEN = 0xB4), and *Table 704* when TrustZone® is disabled (TZEN = 0xC3).

Table 703. Authentication signal states with TrustZone® enabled (TZEN = 0xB4)

Debug state	Authentication signal state	Description
OPEN	dbgen = 1 spiden = 1 niden = 1 spniden = 1	Debug and trace is enabled whatever the state of the processor. Debugger access to secure memory is permitted.

Table 703. Authentication signal states with TrustZone® enabled (TZEN = 0xB4)

Debug state	Authentication signal state	Description
CLOSED SECURE	dbgen = 1 spiden = 0 niden = 1 spniden = 0	Debug and trace is enabled when the processor is in non-secure state. Debugger access to non-secure memory is permitted. Debugger access to secure memory is disabled.
CLOSED	dbgen = 0 spiden = 0 niden = 0 spniden = 0	Debug and trace is disabled.

Table 704. Authentication signal states with TrustZone® disabled (TZEN = 0xC3)

Debug state	Authentication signal state	Description
OPEN	dbgen = 1 spiden = 1 niden = 1 spniden = 1	Debug and trace is enabled whatever the state of the processor. All memory and resources are considered non-secure and accessible to the debugger.
CLOSED	dbgen = 0 spiden = 0 niden = 0 spniden = 0	Debug and trace is disabled.

The state of the authentication signals can be read from the DAUTHSTATUS register in the system control space (SCS) of the Cortex®-M33 processor.

The debug state depends on the state of the product life cycle and debug authentication (see *Section 60.2.7*).

Table 705. Life cycle state and debug states

Product life cycle state	Debug state
OPEN	OPEN
TZ_CLOSED	CLOSED-SECURE
CLOSED (debug not authenticated)	CLOSED
CLOSED (debug constrained)	OPEN/CLOSED-SECURE(1)
LOCKED	CLOSED

1. Depends on authorization level.

60.2.7 Debug authentication

Figure 963. Product life cycle states and debug authentication

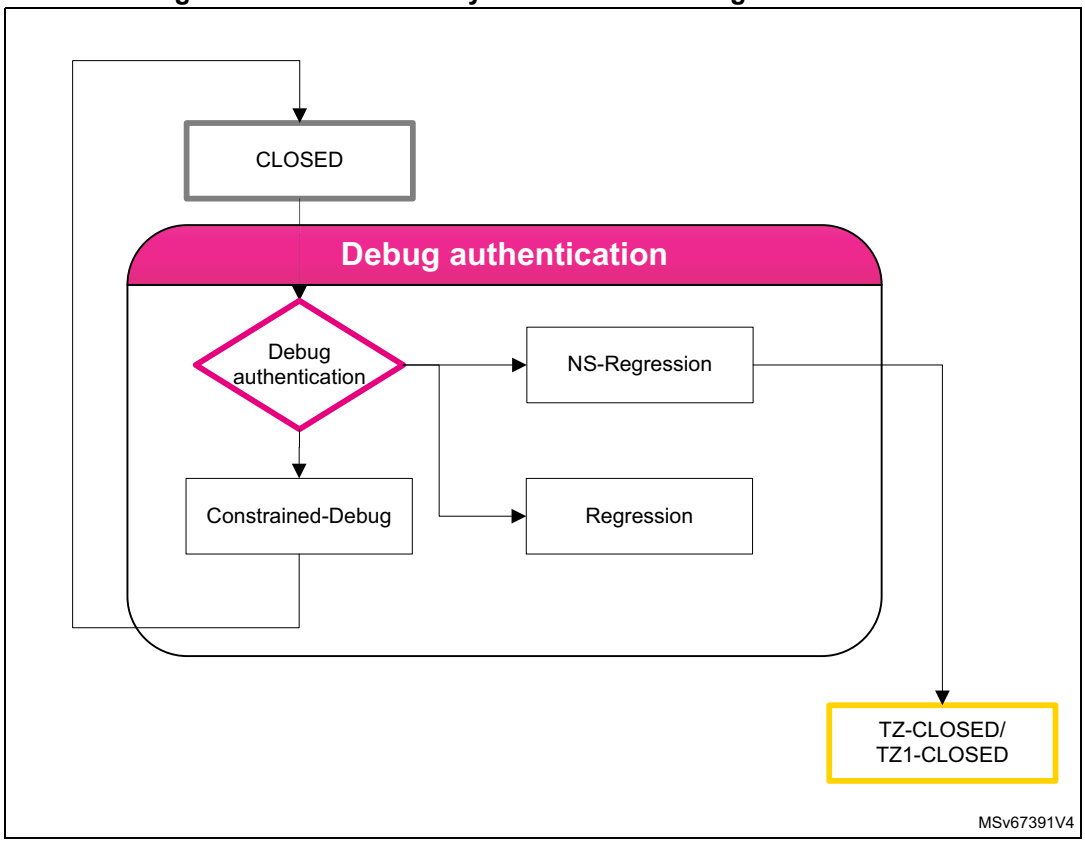


Figure 963 shows the product life cycle and debug authentication states.

If the device is in CLOSED (product life cycle) state, the debug state is CLOSED. The debug authentication procedure allows a trusted debugger to reopen access without compromising sensitive information called the Root Of Trust (ROT).

Reopening the debug is possible only if sensitive asset security is ensured (and TrustZone® is enabled (TZEN = 0xB4). This is called constrained debug, as constraints ensure the security of the ROT information.

Alternatively, a partial or full regression mechanism can be used when security of sensitive information cannot be guaranteed. This is called regression, as regression ensures removal of the sensitive information before reopening the debug.

- Partial regression corresponds to releasing nonsecure code and assets. The intermediate state, which allows partial regression management, is called NS-regression.
- Full regression corresponds to releasing all code and assets. The intermediate state allowing full regression management is called regression.

Debug authentication control principle

The debug authentication is one of the most critical security features of the system considering that with a debugger the user can access a large part of the system.

To control the reopening of the debug, the device imposes a debug authentication protocol.

The protocol implements a challenge response mechanism based on asymmetric cryptography to authenticate the host. It relies on a key pair, with a public key stored in the device, and a private key from the host library, used to sign a random value (the challenge) generated by the device.

The protocol implements a bidirectional communication between the host and the device through a mailbox interface located in the DBGMCU.

The host can write to the mailbox via the JTAG/SWD interface. It expects to get responses and messages from the device via the same mailbox.

The debug authentication protocol is launched on a power-on reset of the device, when an “open request” message is posted by the host.

The protocol is based on:

- Initial message: posted by the host combined with a reset to launch the debug authentication process on the device.
- Challenge message: the device generates a random value, to be signed by the host, when sending back the response.
- Response: the host sends a message to the device proving its authenticity. This is done using a tool to generate a token.

The implementation is ensured by code embedded in the system flash memory. This code is called automatically after reset if an initial message has been posted by the host in the mailbox.

After a first sequence of mutual authentication to align on protocol version, type of device, and similar parameters, the device generates a random value that is signed by the host with a private key when building the response.

The STMicroelectronics implementation is based on the Arm® PSA-ADAC solution for debug authentication.

The debug authentication can be implemented using a proprietary or open-source protocol.

As this feature is critical for security, the STM32G56x/55x/54x/53x devices come with debug authentication provisioned in the system flash. STMicroelectronics provides the host tools integrated with some debug environments (IDEs).

Debug authentication provisioning

Debug authentication is natively supported by the STM32G56x/55x/54x/53x platform. It means that the data used by ST debug authentication (ST-DA) must be provisioned at a defined location in the secure key storage area (OBKeys defined in flash memory).

The debug authentication configuration must be done only when the PRODUCT_STATE is “Provisioning”, and cannot be performed when the PRODUCT_STATE is “Open”.

The following data must be provisioned (refer to Section 1.7.2: RSS user functions):

- Data must be provisioned at the very beginning of the OBK_HDPL1 mapping (0x0FFD 0100) (see Section 1.5.2: OBK access per HDPL level)
- RSS_Lib encryption option must be set for STM32G56x/STM32G54x devices
- RSS_Lib encryption option must be reset for STM32G55x/STM32G53x devices

The data to provision are defined in *Table 666* and are dependent on the TZEN option byte setting (enabled or disabled).

Table 706. Definition of data to provision

Data size	Content	Comment
32 bytes	SHA256 of the overall blob (i.e. the next fields)	-
32 bytes	SHA256 of the ROT certificate public key for Key Type = Certificate or SHA256 of the password for Key Type = Password	-
64 bytes	Reserved	-
4 bytes	MAGIC	“STDA”
1 byte	Key Type	0x01: Certificate 0x80: Password
43 bytes	reserved	-
16 bytes	128-bits permission mask (refer to *Table 707*) for Key Type = Certificate or Reserved for Key Type = Password	-

Caution: Configure debug authentication as explained above before changing to states different from Open or Provisioning.

Table 707. Permission mask (Endianness: Little Endian)

Bit(s)	Description
Bit 0	When set it allows Intrusive debug from HDPL1 – NS
Bit 1	When set it allows Intrusive debug from HDPL2 – NS
Bit 2	When set it allows Intrusive debug from HDPL3 – NS
Bit 3	Reserved
Bit 4	When set it allows Intrusive debug from HDPL1 – S
Bit 5	When set it allows Intrusive debug from HDPL2 – S
Bit 6	When set it allows Intrusive debug from HDPL3 – S
Bits 7:11	Reserved
Bit 12	Regression to TZ-Closed, when 1 regression to TZ-Closed is allowed
Bit 13	Reserved
Bit 14	Regression (Full regression), when 1 full regression is allowed
Bit 15:125	Reserved
Bit 126	Dry run
Bit 127	Reserved

60.3 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight™ component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- A 5-pin standard JTAG interface (JTAG-DP)
- A 2-pin (clock + data) serial-wire debug port (SW-DP)

These two modes are mutually exclusive, since they share the same IO pins.

By default, the JTAG-DP is selected after a system or a power-on reset. The five IO pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

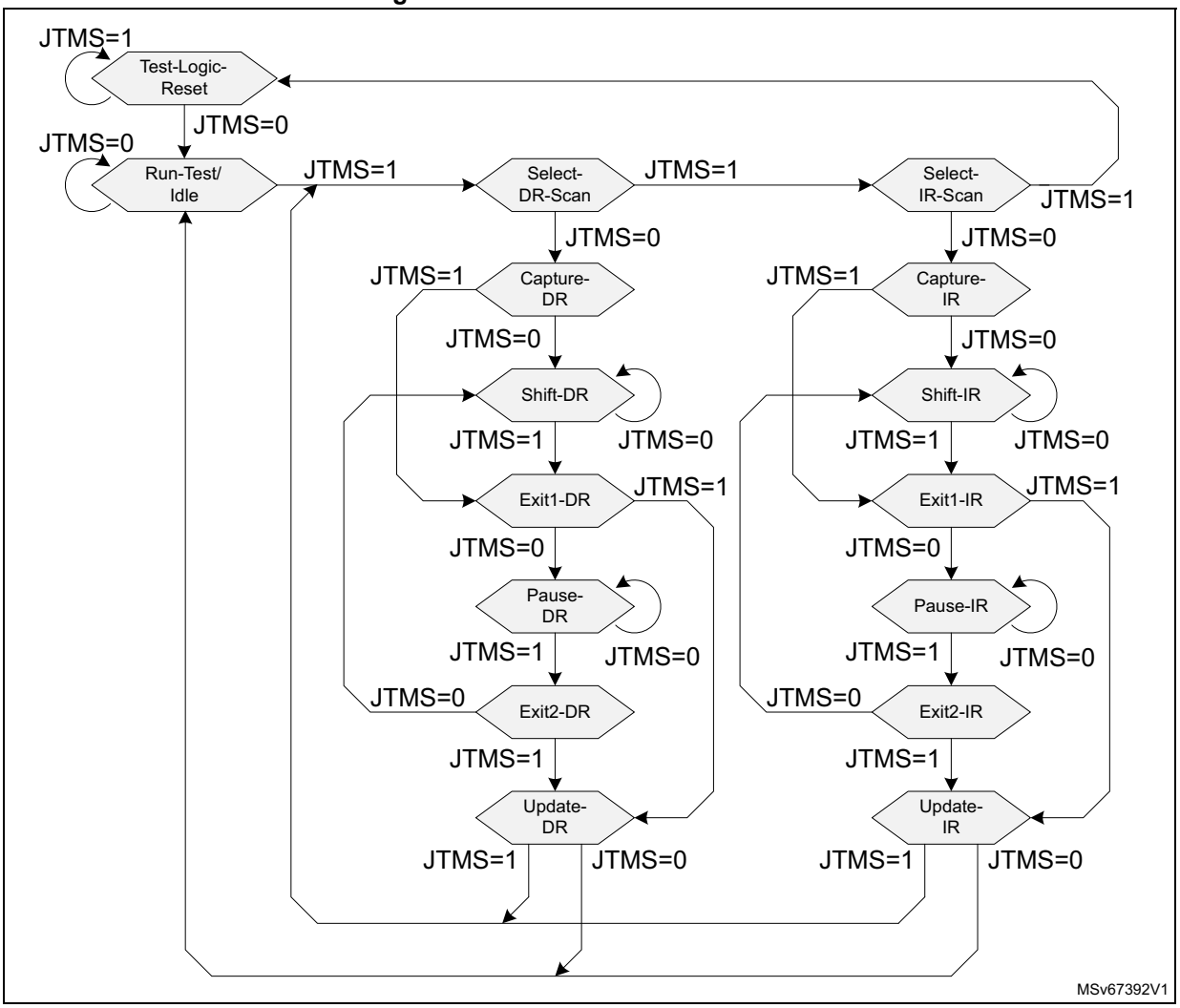
In SW-DP mode, the unused JTAG pins JTDI, JTDO, and nJTRST can be used for other functions.

Note: All SWJ port IOs can be reconfigured for other functions by software, but debugging is then no longer possible.

60.3.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM), shown in the figure below, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR) and the other one with a number of data registers (DR).

Figure 964. JTAG TAP state machine



The operation of the JTAG-DP is as follows:

1. When the TAPSM goes through the Capture-IR state, 0b0001 is transferred to the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
2. While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
   – The LSB of the IR scan chain is output on JTDO.
   – Bit[n] of the IR scan chain is transferred to bit[n-1].
   – The value on JTDI is transferred to the MSB of the IR scan chain.
3. When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred to the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers to one of the DR scan chains, connected between JTDI and JTDO.
5. The value held in the instruction register determines which data register and associated DR scan chain are selected.
6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shifts in the Shift-IR state.
7. When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred to the selected data register.
8. When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in the IR.

When active, the nJTRST signal resets the state machine asynchronously to the test-logic-reset state.

The data registers corresponding to the 4-bit IR instructions are listed in the table below.

Table 708. JTAG-DP data registers

IR instruction	DR register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – bits 34:3 = APABORT[31:0]: Write 0x0000 0001 to abort an ongoing access port transaction – bits 2:1 = A[3:2] = 00 – bit 0 = RnW = 0: write only
1001	(BYPASS)	1	Reserved: BYPASS selected

Table 708. JTAG-DP data registers (continued)

IR instruction	DR register	Scan chain length	Description
1010	DPACC	35	Debug port access register Initiates the debug port and gives access to a debug port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request bits 2:1 = A[3:2] = 2-bit address of a debug port register bit 0 = RnW = read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: - 010 = OK/FAULT - 001 = WAIT - others = reserved
1011	APACC	35	Access port access register Initiates an access port and gives access to an access port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request bits 2:1 = A[3:2] = 2-bit sub-address of an access port register bit 0 = RnW= Read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: - 010 = OK/FAULT - 001 = WAIT - others= reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	Identification code 0x6BA0 0477: Cortex®-M33 JTAG debug port ID code
1111	BYPASS	1	Bypass A single JTCK cycle delay is inserted between JTDI and JTDO.

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification (see *Section 60.13*).

60.3.2 Serial-wire debug port

The serial-wire debug protocol uses the following pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock.

A transfer comprises three phases:

1. Packet request (8 bits) transmitted by the host, see *Table 709*
2. Acknowledge response (3 bits) transmitted by the target, see *Table 710*
3. Data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read), see *Table 711*

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock-cycle turn-around time is inserted.

Table 709. Packet request

Bit field	Name	Description
0	Start	Must be 1
1	APnDP	– 0: DP register access - see *Section 60.3.3* – 1: AP register access - see *Section 60.4*
2	RnW	– 0: write request – 1: read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 713* or *Table 715*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by target

Table 710. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 711. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single-bit parity of 32 data bits

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case, the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the serial-wire debug protocol, refer to the Arm® Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

60.3.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in *Table 713*.

The debugger can access the DP registers as follows:

1. Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the data field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit reset (see *Table 709*). The write data are sent in the data phase.
2. To access one of the banked DP registers at address 0x4, the register number must first be written in the DP_SELECTR register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the contents of the DP_SELECTR register.

Table 712. Debug port registers

Address	A(3:2) value	R/W	Description
0x0	00	R	*DP debug port identification register (DP_DPIDR)* contains the IDCODE for the debug port.
		W	*DP abort register (DP_ABORTR)*(1) aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRLSTATR register.
0x4	01	R/W	If DP_SELECTR.DPBANKSEL[3:0] = 0x0, *DP control and status register (DP_CTRLSTATR)* controls the DP and provides status information.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x1, *DP data link control register (DP_DLCR)*(2) controls the operating mode of the SWD data link.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x2, *DP target identification register (DP_TARGETIDR)* provides target identification information.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x3, *DP data link protocol identification register (DP_DLPIDR)*(2) provides the SWD protocol version.
0x8	10	R	*DP event status register (DP_RESENDR)*(2) returns the value that was returned by the last AP read or DP_RDBUFF read. Used in the event of a corrupted read transfer.
		W	*DP access port select register (DP_SELECTR)* selects the access port, access port register bank, and DP register at address 0x4.
0xC	11	R	*DP read buffer register (DP_RDBUFFR)* – Via JTAG-DP, this register allows the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation) – Via SW-DP, this register contains the result of the preceding AP read access, allowing a new AP access to be avoided.

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x6BA0 2477

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]**: revision code
0x6: Rev 6

Bits 27:20 **PARTNO[7:0]**: part number for the debug port
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: minimal debug port (MINDP) implementation
0x0: MINDP not implemented

Bits 15:12 **VERSION[3:0]**: debug port architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

DP abort register (DP_ABORTR)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
											w	w	w		w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: overrun error clear
0: no effect
1: STICKYORUN bit cleared in DP_CTRL/STATR register

Bit 3 **WDERRCLR**: write data error clear
0: no effect
1: WDATAERR bit cleared in DP_CTRL/STATR register

Bit 2 **STKERRCLR**: sticky error clear
0: no effect
1: STICKYERR bit cleared in DP_CTRL/STATR register

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT**: current AP transaction aborted if excessive number of WAIT responses returned
This bit indicates that the transaction is stalled.
0: no effect
1: transaction aborted

DP control and status register (DP_CTRLSTATR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
								r	r	r				r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK**: debug power-up acknowledge
See description in *Section 60.2.5: Debug and low-power modes*.
0: DCLK gated
1: DCLK enabled

Bit 28 **CDBGPWRUPREQ**: debug power-up request
This bit controls the DCLK enable request signal.
0: requests DCLK gating
1: requests DCLK enable

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR**: write data error (read-only) in SW-DP

This bit indicates that there is a parity or framing error on the data phase of a write, or a write accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to the ABORT.WDERRCLR bit.

0: no error

1: an error occurred

Note: This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response (read-only) in SW-DP

This bit indicates the response to the last AP read access.

0: read not OK

1: read OK

Note: This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: transaction error (read-only in SW-DP, read/write in JTAG-DP)

This bit indicates that an error occurred in an AP transaction.It is reset by writing 1 to the DP_ABORTR.STKERRCLR bit (in SW-DP and JTAG-DP)

0: no error

1: an error occurred

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN**: overrun (read-only in SW-DP, read/write in JTAG-DP).

This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set. it is reset by writing 1 to the DP_ABORTR.ORUNERRCLR bit (in SW-DP and JTAG-DP).

0: no overrun

1: an overrun occurred

Bit 0 **ORUNDETECT**: overrun detection mode enable.

0: disabled

1: enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

DP data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
						r	r	r	r						

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: tristate period for SWDIO
0x0: 1 data bit period

Bits 7:6 **WIREMODE[1:0]**: SW-DP mode
0x0: synchronous mode

Bits 5:0 Reserved, must be kept at reset value.

DP target identification register (DP_TARGETIDR)

Address offset: 0x4

Reset value: 0xXXXX 0041

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]**: target revision

Bits 27:12 **TPARTNO[15:0]**: target part number
0x44A0: STM32G55x/56x
0x44B0: STM32G53x/54x

Bits 11:1 **TDESIGNER[10:0]**: target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: target instance number
this field defines the instance number for the device in a multi-drop system.
0x0: instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial-wire debug protocol version
0x1: version 2

DP event status register (DP_RESENDR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]**: value returned by the last AP read or DP_RDBUFF read
This register is used in the event of a corrupted read transfer.

DP access port select register (DP_SELECTR)

Address offset: 0x8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: access port select
This field selects the access port for the next transaction.
0x00: AP0 - System debug access port (APB-AP)
0x01: AP1 - Cortex®-M33 debug access port (AHB-AP)
Others: reserved

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select
This field selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select

This field selects the register at address 0x4 of the debug port.

0x0: DP_CTRLSTAT register

0x1: DP_DLCR register

0x2: DP_TARGETID register

0x3: DP_DLPIDR register

Others: reserved

DP read buffer register (DP_RDBUFFR)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: value returned by the last AP read access

The value returned by an AP read access can be obtained using a second read access to the same address (initiates a new transaction on the corresponding bus), or can be read from this register, in which case no new AP transaction occurs.

60.3.4 Debug port register map and reset values

These registers are not on the CPU memory bus. They are accessed only through the SW-DP and JTAG-DP debug interface.

The debug port address offset is 4-bit wide, where the two most significant bits are defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field. The two least significant bits are 00.

Table 713. Debug port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	**DP_DPIDR**	REVISION [3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION [3:0]				DESIGNER[10:0]											Res.
	Reset value	0	1	1	0	1	0	1	1	1	0	1	0				0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1
0x0	**DP_ABORTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
	Reset value																												0	0	0		0

Table 713. Debug port register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4(1)	**DP_CTRLSTATR**	Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
	Reset value			0	0																					0	0	0				0	0
0x4(2)	**DP_DLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0	0	0						
0x4(3)	**DP_TARGETIDR**	TREVISION [3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
0x4(4)	**DP_DLPIDR**	TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
	Reset value	0	0	0	0																									0	0	0	1
0x8	**DP_RESENDR**	RESEND[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	**DP_SELECTR**	APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
	Reset value	x	x	x	x	x	x	x	x																	x	x	x	x	x	x	x	x
0xC	**DP_RDBUFFR**	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DP_SELECTR.DPBANKSEL[3:0] = 0x0.
2. DP_SELECTR.DPBANKSEL[3:0] = 0x1.
3. DP_SELECTR.DPBANKSEL[3:0] = 0x2.
4. DP_SELECTR.DPBANKSEL[3:0] = 0x3.

60.4 Access ports

There are two access ports (AP) attached to the DP.

- System debug access port (AP0): enables access to the DBGMCU and the system ROM table via an APB bus.
- Cortex®-M33 debug access port (AP1): enables access to the debug and trace features integrated in the Cortex®-M33 processor core via its internal AHB bus.

60.4.1 Access port registers

The access ports are of type MEM-AP: the debug and trace component registers are mapped in the address space of the AHB. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 715*.

The address of the AP registers is composed of the following fields:

- bits [7:4]: content of the APBANKSEL[3:0] field in the *DP access port select register (DP_SELECTR)*)
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 708*), or of the SW-DP packet request (see *Table 709*), depending on the debug interface used
- bits [1:0]: always set to 0

The content of the DP_SELECTR.APSEL[3:0] field defines which MEM-AP is being accessed.

Table 714. MEM-AP registers

Address	APBANKSEL	A(3:2)	Name	Description
0x00	0x0	0	CSWR	Control/status word register
0x04	0x0	1	TAR	Transfer address register Target address for the bus transaction.
0x08	-	-	-	Reserved
0x0C	0x0	3	DRWR	Data read/write register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0]
0x10	0x1	0	BD0R	Banked data 0 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x0.
0x14	0x1	1	BD1R	Banked data 1 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x4.
0x18	0x1	2	BD2R	Banked data 2 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x8.
0x1C	0x1	3	BD3R	Banked data 3 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0xC.
0x20	-	-	-	Reserved
0x24 to 0xEC	-	-	-	Reserved
0xF0	-	-	-	Reserved
0xF4	0xF	1	CFGR	Configuration register (read only)
0xF8	0xF	2	BASER	Debug base address register (read only) Base address of the ROM table

Table 714. MEM-AP registers (continued)

Address	APBANKSEL	A(3:2)	Name	Description
0xFC	0xF	3	IDR	Identification register (read only)

The debugger can access the AP registers as follows:

1. Program the APSEL[3:0] field in the *DP access port select register (DP_SELECTR)* to choose the AP, and the APBANKSEL[3:0] field in DP_SELECTR to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC data register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 709*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the *APx transfer address register (APx_TAR) (x = 0, 1)*.
2. Program the *AP1 control/status word register (AP1_CSWR)*, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the *APx data read/write register (APx_DRWR) (x = 0, 1)* to initiate a bus transaction at the address held in AP_TAR. Alternatively, a read or write to the *APx banked data n register (APx_BDnR) (x = 0, 1)* triggers access to TAR[31:4] + n address, allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register.

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification *[1]*.

AP0 control/status word register (AP0_CSWR)

Address offset: 0x0

Reset value: 0x8000 0042

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBGSWEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		r	r	r

Bit 31 **DBGSWEN**: software access enable
Enables or disables software access to the APB bus
0: disable software access
1: enable software access

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: mode of operation
0b0000: normal download or upload
other: reserved

Bit 7 **TRINPROG**: transfer in progress (read only)
This field indicates whether a transfer is currently in progress on the APB master port
0: no APB transfer in progress
1: APB transfer in progress

Bit 6 **DEVICEEN**: device enable status (read only)
1: APB transfers always enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size in bytes of the transaction (SIZE field)
other: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction
0x2: word (32-bit)

AP1 control/status word register (AP1_CSWR)

Address offset: 0x0

Reset value: 0x43X0 00X2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPROT	Res.	Res.	PROT[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw	rw	rw	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
								r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: secure transfer request
This field sets the protection attribute HPROT[6] of the bus transfer.
0: secure transfer, HPROT[6] = low
1: non-secure transfer, HPROT[6] = high
If SPIDEN = 0 and SPROT = 0, no bus transfer occurs

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **PROT[3:0]**: bus transfer protection
This field sets the protection attributes HPROT[3:0] of the bus transfer.
0bXXX0: instruction access
0bXXX1: data access
0bXX0X: user mode
0bXX1X: privilege mode
0bX0XX: non-bufferable
0bX1XX: bufferable
0b0XXX: non-shareable, no look-up, non-modifiable
0b1XXX: shareable, look-up, modifiable

Bit 23 **SPISTATUS**: secure debug authentication status
This field indicates the state of the SPIDEN signal
0: No secure AHB transfers allowed
1: Secure AHB transfers allowed

Bits 22:8 Reserved, must be kept at reset value.

Bit 7 **TRINPROG**: transfer in progress (read only)
This field indicates whether a transfer is currently in progress on the APB master port
0: No AHB transfer in progress
1: AHB transfer in progress

Bit 6 **DBGSTATUS**: debug enable (DBGEN) status
0: AHB transfers blocked
1: AHB transfers enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size of the transaction (SIZE field), in bytes. Single transfer.
0x2: address incremented by the size of the transaction (SIZE field), in bytes. Packs four 8-bit or two 16-bit transfers into a 32-bit DAP transfer. Multiple transactions are carried out on the AHB interface.
Others: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction
0x0: byte (8-bit)
0x1: halfword (16-bit)
0x2: word (32-bit)
Others: reserved

APx transfer address register (APx_TAR) (x = 0, 1)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TA[31:0]**: address of current transfer. In AP0, TA[1:0] are fixed at 0.

APx data read/write register (APx_DRWR) (x = 0, 1)

Address offset: 0x0C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TD[31:0]**: data of current transfer

APx banked data n register (APx_BDnR) (x = 0, 1)

Address offset: 0x10 + 4 * n, (n = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]:** banked data of current transfer to address TA [31:4] + 4 * n.
Auto address incrementing is not performed on APx_BD0-3R. Banked transfers are only supported for word transfers.

APx base address register (APx_BASER) (x = 0, 1)

Address offset: 0xF8

Reset value: AP 0: 0xE00E 0003

Reset value: AP 1: 0xE00F E003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[31:12]**: base address (bits 31 to 12) of the first ROM table
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
0xE00E0: AP0
0xE00FE: AP1

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: base-address register format
1: Arm® debug interface v5

Bit 0 **ENTRYPRESENT**: debug components presence
Indicates that debug components are present on the access port bus.
1: debug components present

APx identification register (APx_IDR) (x = 0, 1)

Address offset: 0xFC

Reset value: AP 0: 0x5477 0002

Reset value: AP 1: 0x8477 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3]
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLASS[2:0]			Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
r	r	r						r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: revision number
0x5: r1p0
0x8: r0p9

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm®

Bits 16:13 **CLASS[3:0]**:
0x1: MEM-AP

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**:
0x1: AHB-AP
0x2: APB-AP

60.4.2 Access port register map

These registers are not on the CPU memory bus. They are only accessed through the SW-DP and JTAG-DP debug interfaces.

The access port address is 8-bit wide, defined by DP_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 715. Access port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**AP0_CSWR**	DBGSWEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value	1																				0	0	0	0	0	1	0	0		0	1	0
0x00	**AP1_CSWR**	Res.	SPROT	Res.	Res.	PROT[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		1			0	0	1	1	X												0	0	0	0	0	X	0	0		0	1	0
0x04	**APx_TAR**	TA[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	Reserved	Reserved																															
0x0C	**APx_DRWR**	TD[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x10	**APx_BD0R**	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**APx_BD1R**	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**APx_BD2R**	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**APx_BD3R**	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20 to 0xF4	Reserved	Reserved																															
0xF8	**AP0_BASER**	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0										1	1

Table 715. Access port register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF8	**AP1_BASER**	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0											1	1
0xFC	**AP0_IDR**	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	0	1	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	0	0	0	1	0
0xFC	**AP0_IDR**	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	0	0	0	0	1

60.5 ROM tables

The ROM table is a CoreSight™ component that contains the base addresses of the CoreSight™ debug components accessible via the access port to which it is attached. These tables allow a debugger to discover the topology of the CoreSight™ system automatically.

There is one top-level ROM table behind each access port, APn. The base address of this ROM table can be obtained by reading the APn_BASER register of the access port. The top-level ROM table may point in turn to other ROM tables.

The system ROM table is pointed to by the AP0 base register, AP0_BASER. It contains the base address pointer for the DBGMCU.

The system ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00E 0000 to 0xE00E 0FFC, when accessed by the debugger. It can be accessed by the CPU at the address range 0x4402 0000 to 0x4402 0FFC.

Table 716. System ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	DBGMCU	0xE00E 6000 (debugger) 0x4402 6000 (CPU) 0x5402 6000 (CPU secure)	0x0000 6000	4	0x0000 6003
0x004	Top of table	-	-	-	0x0000 0000

Table 716. System ROM table (continued)

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x008 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 719*

There are two ROM tables in the CPU subsystem. The MCU ROM table is pointed to by the AP1 base register, AP1_BASER. It contains the base-address pointer for the processor ROM table and for the TPIU registers.

The MCU ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFC.

Table 717. MCU ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	Processor ROM table	0xE00F F000	0x0000 1000	4	0x0000 1003
0x004	TPIU	0xE004 0000	0xFFF4 2000	4	0xFFF4 2003
0x008	Reserved	-	-	-	0x1FF0 2002
0x00C	Reserved	-	-	-	0x1FF0 2002
0x010	Top of table	-	-	-	0x0000 0000
0x014 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 720*

The processor ROM table contains the base-address pointer for the system control space (SCS) registers, that allow the debugger to identify the CPU core, as well as for the BPU, DWT, ITM, ETM, and CTI.

The processor ROM table (see *Table 718*) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

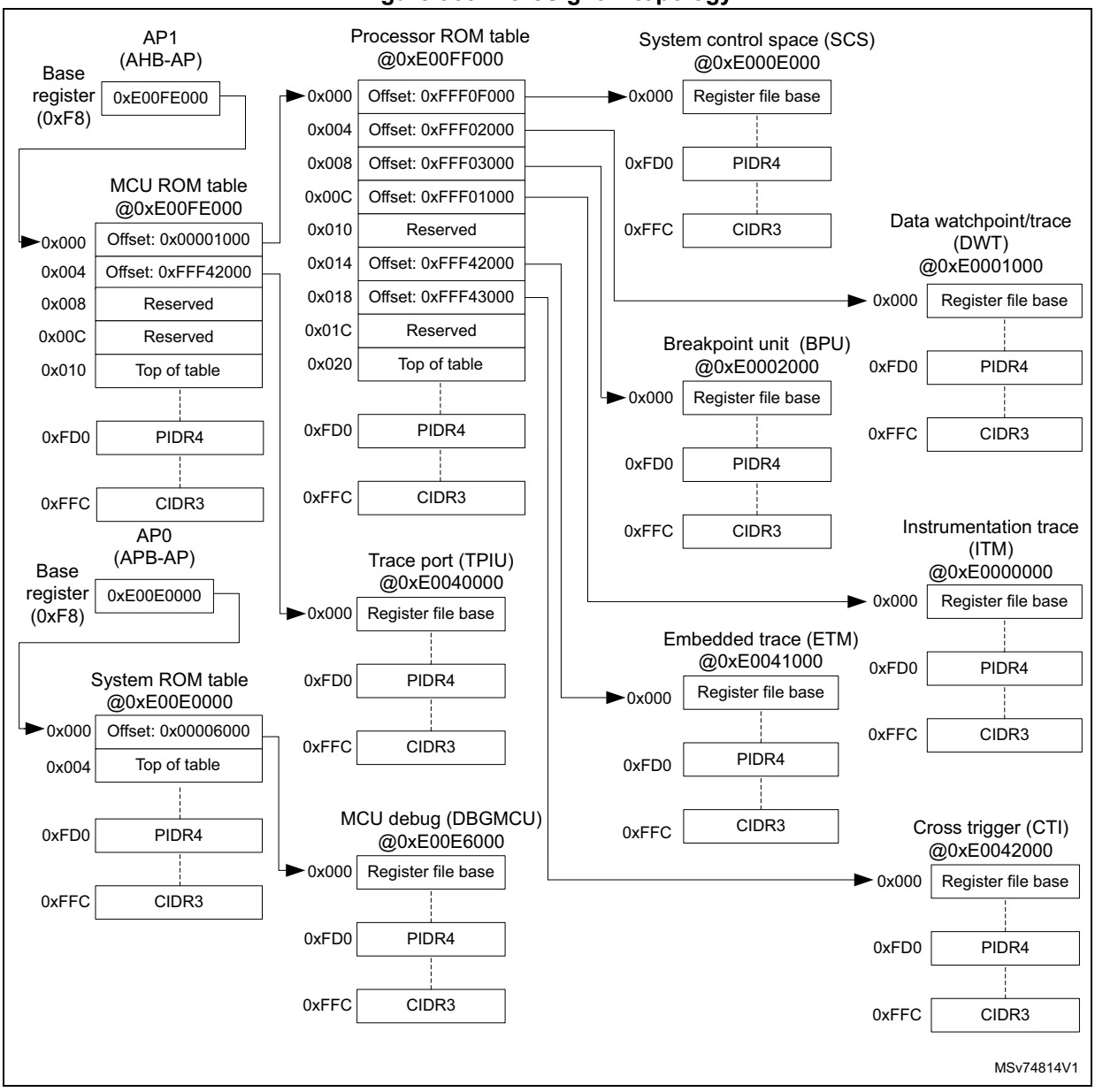
Table 718. Processor ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F000	SCS	0xE000 E000	0xFFF0 F000	4	0xFFF0 F003
0xE00F F004	DWT	0xE000 1000	0xFFF0 2000	4	0xFFF0 2003
0xE00F F008	BPU	0xE000 2000	0xFFF0 3000	4	0xFFF0 3003
0xE00F F00C	ITM	0xE000 0000	0xFFF0 1000	4	0xFFF0 1003
0xE00F F010	Reserved	-	-	-	0xFFF4 1002
0xE00F F014	ETM	0xE004 1000	0xFFF4 2000	4	0xFFF4 2003
0xE00F F018	CTI	0xE004 2000	0xFFF4 3000	4	0xFFF4 3003
0xE00F F01C	Reserved	-	-	-	0xFFF4 4002

Table 718. Processor ROM table (continued)

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F020	Top of table	-	-	-	0x0000 0000
0xE00F F024 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00F FFCC to 0xE00F FFFC	ROM table registers	-	-	-	See *Table 721*

Figure 965. CoreSight™ topology



60.5.1 System ROM table registers

System ROM memory type register (SYSROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
0x1: system memory present on this bus

System ROM CoreSight™ peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

System ROM CoreSight™ peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x4A: STM32G55x/56x
0x4B: STM32G53x/54x

System ROM CoreSight™ peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: STM32G5xx

System ROM CoreSight™ peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

System ROM CoreSight™ peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

System ROM CoreSight™ component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: CoreSight™ component identification bits [7:0]
0x0D: Common identification value

System ROM CoreSight™ peripheral identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

System ROM CoreSight™ component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

System ROM CoreSight™ component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

60.5.2 System ROM table register map

Table 719. System ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	**SYSROM_ MEMTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	**SYSROM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to FDC	Reserved	Reserved																															
0xFE0	**SYSROM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0xFE4	**SYSROM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	X	X	X	X
0xFE8	**SYSROM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	**SYSROM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**SYSROM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**SYSROM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	**SYSROM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**SYSROM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 716* for register boundary addresses.

60.5.3 MCU ROM table registers

MCU ROM memory type register (MCUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
0x1: system memory present on this bus

MCU ROM CoreSight™ peripheral identity register 4 (MCUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

MCU ROM CoreSight™ peripheral identity register 0 (MCUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x4A: STM32G55x/56x
0x4B: STM32G53x/54x

MCU ROM CoreSight™ peripheral identity register 1 (MCUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: STM32G5xx

MCU ROM CoreSight™ peripheral identity register 2 (MCUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

MCU ROM CoreSight™ peripheral identity register 3 (MCUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

MCU ROM CoreSight™ component identity register 0 (MCUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: Common identification value

MCU ROM CoreSight™ peripheral identity register 1 (MCUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

MCU ROM CoreSight™ component identity register 2 (MCUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

MCU ROM CoreSight™ component identity register 3 (MCUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

60.5.4 MCU ROM table register map

Table 720. MCU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	**MCUROM_ MEMTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	**MCUROM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to FDC	Reserved	Reserved																															
0xFE0	**MCUROM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0xFE4	**MCUROM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	X	X	X	X
0xFE8	**MCUROM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	**MCUROM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**MCUROM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**MCUROM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	MCUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	MCUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

60.5.5 Processor ROM table registers

CPU ROM memory type register (CPUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
1: system memory present on this bus

CPU ROM CoreSight™ peripheral identity register 4 (CPUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: ARM® JEDEC continuation code

CPU ROM CoreSight™ peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0xC9: Cortex®-M33

CPU ROM CoreSight™ peripheral identity register 1 (CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: ARM® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: Cortex®-M33

CPU ROM CoreSight™ peripheral identity register 2 (CPUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CPU ROM CoreSight™ peripheral identity register 3 (CPUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

CPU ROM CoreSight™ component identity register 0 (CPUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

CPU ROM CoreSight™ peripheral identity register 1 (CPUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

CPU ROM CoreSight™ component identity register 2 (CPUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

CPU ROM CoreSight™ component identity register 3 (CPUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

60.5.6 Processor ROM table register map

Table 721. CPU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	**CPUROM_ MEMTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD4 to FDC	Reserved	Reserved																															
0xFD0	**CPUROM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	**CPUROM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	1	0	0	1	0	0	1
0xFE4	**CPUROM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	**CPUROM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	**CPUROM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**CPUROM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**CPUROM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	**CPUROM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**CPUROM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

60.6 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following:

- Watchpoint
- ETM trigger
- PC sampling trigger
- Data address sampling trigger
- Data comparator (COMP 1 only)
- Clock cycle counter comparator (COMP 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its *DWT comparator x register (DWT_COMPxR)* with one of the following:

- A data address
- An instruction address
- A data value
- The cycle-count value, for COMP 0 only

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT data trace packets, containing one or more of:
  - The address of the instruction that caused a data access
  - An address offset, bits[15:0] of the data access address
  - The matched data value
- A watchpoint debug event, on either the PC value or the accessed data address
- A CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter debug state.

For more details on how to use the DWT, refer to the Arm®v8-M Architecture Reference Manual *[4]*.

60.6.1 DWT registers

The DWT registers are located at the address range 0xE000 1000 to 0xE000 1FFC.

DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: number of comparators implemented (read only)
0x4: four comparators

Bit 27 **NOTRCPKT**: trace sampling and exception tracing support (read only)
0: supported

Bit 26 **NOEXTTRIG**: external match signal, CMPMATCH support (read only)
0: supported

Bit 25 **NOCYCCNT**: cycle counter support (read only)
0: supported

Bit 24 **NOPRFCNT**: profiling counter support (read only)
0: supported

Bit 23 **CYCDISS**: cycle counter disabled secure.
Controls whether the cycle counter is disabled in secure mode.
0: no effect
1: disable incrementing of the cycle counter when the processor is in secure state

Bit 22 **CYCEVTENA**: enable for POSTCNT underflow event counter packet generation
0: disabled
1: enabled

Bit 21 **FOLDEVTENA**: enable for folded instruction counter overflow event generation
0: disabled
1: enabled

Bit 20 **LSUEVTENA**: enable for LSU counter overflow event generation
0: disabled
1: enabled

Bit 19 **SLEEPEVTENA**: enable for sleep counter overflow event generation
0: disabled
1: enabled

Bit 18 **EXCEVTENA**: enable for exception overhead counter overflow event generation
0: disabled
1: enabled

Bit 17 **CPIEVTENA**: enable for CPI counter overflow event generation
0: disabled
1: enabled

Bit 16 **EXCTRCENA**: enable for exception trace generation
0: disabled
1: enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: enable for POSTCNT counter to be used as a timer for periodic PC sample packet generation
0: disabled
1: enabled

Bits 11:10 **SYNCTAP[1:0]**: position of the synchronization packet counter tap on the CYCCNT counter
This field determines the synchronization packet rate.
00: disabled, no synchronization packets
01: Tap at CYCCNT[24]
10: Tap at CYCCNT[26]
11: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Selects the position of the POSTCNT tap on the CYCCNT counter.
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: initial value of the POSTCNT counter
Writes to this field are ignored if POSTCNT counter is enabled. CYCEVTENA or PCSAMPLENA bits must be reset prior to writing POSTINIT.

Bits 4:1 **POSTRESET[3:0]**: reload value of the POSTCNT counter

Bit 0 **CYCCNTENA**: enable CYCCNT counter
0: disabled
1: enabled

DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CYCCNT[31:0]**: processor clock-cycle counter

DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter

Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNTR, and counts any instruction fetch stalls.

DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: exception overhead cycle counter
Counts the number of cycles spent in exception processing.

DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: load store counter
Counts additional cycles required to execute load and store instructions.

DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: folded instruction counter
Increments on each instruction that takes 0 cycles.

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EIASAMPLE[31:0]**: executed instruction address sample value.
Samples the current value of the program counter.

DWT comparator x register (DWT_COMPxR)

Address offset: 0x020 + 0x010 * x, (x = 0 to 3)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COMP[31:0]**: reference value for comparison

DWT function register 0 (DWT_FUNCTR0)

Address offset: 0x028

Reset value: 0x5800 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for match for comparator 0.
0b01011: Cycle Counter, Instruction Address, Data Address and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 0.
For possible values of this field, refer to *[4]*.

DWT function register 1 (DWT_FUNCTR1)

Address offset: 0x038

Reset value: 0xD000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for match for comparator 1.
0b11010: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 1.
For possible values of this field, refer to *[4]*.

DWT function register 2 (DWT_FUNCTR2)

Address offset: 0x048

Reset value: 0x5000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2
0b01010: Instruction Address, Data Address, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size:
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*

DWT function register 3 (DWT_FUNCTR3)

Address offset: 0x058

Reset value: 0xF000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2.
0b11110: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, Data value, Linked Data Value, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*

DWT device type architecture register (DWT_DEVARCHR)

Address offset: 0xFC8

Reset value: 0x4770 1A02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DWT_DEVARCH register present
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: DWT architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: DWT architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA02: DWT architecture

DWT device type register (DWT_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: sub-type
0x0: other

Bits 3:0 **MAJOR[3:0]**: major type
0x0: miscellaneous

DWT CoreSight™ peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

DWT CoreSight™ peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: Cortex®-M33 DWT part number

DWT CoreSight™ peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: Cortex®-M33 DWT part number

DWT CoreSight™ peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

DWT CoreSight™ peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

DWT CoreSight™ component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: Common identification value

DWT CoreSight™ peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

DWT CoreSight™ component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

DWT CoreSight™ component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

60.6.2 DWT register map

The DWT registers are located at the address range 0xE000 1000 to 0xE000 1FFC.

Table 722. DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DWT_CTRLR**	NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**DWT_CYCCNTR**	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**DWT_CPICNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x00C	**DWT_EXCCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x010	**DWT_SLPCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	**DWT_LSUCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x018	**DWT_FOLDCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x01C	**DWT_PCSR**	EIASAMPLE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 722. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	**DWT_COMP0R**	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x024	Reserved	Reserved																															
0x028	**DWT_FUNCTR0**	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	1			0													0	0					0	0	0	0	0	0
0x02C	Reserved	Reserved																															
0x030	**DWT_COMP1R**	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x034	Reserved	Reserved																															
0x038	**DWT_FUNCTR1**	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	1	1	0	1	0			0													0	0					0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	**DWT_COMP2R**	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x044	Reserved	Reserved																															
0x048	**DWT_FUNCTR2**	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	**DWT_COMP3R**	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x054	Reserved	Reserved																															
0x058	**DWT_FUNCTR3**	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	1	1	1	1	0			0													0	0					0	0	0	0	0	0
0x05C to 0xFC4	Reserved	Reserved																															
0xFC8	**DWT_DEVARCHR**	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHVER[3:0]				ARCHPART[11:0]												
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0
0xFCC	**DWT_DEVTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 722. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	**DWT_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	**DWT_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	**DWT_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	**DWT_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	**DWT_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**DWT_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**DWT_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	**DWT_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**DWT_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

60.7 Instrumentation trace macrocell (ITM)

The ITM generates trace information in packets. Three sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources in decreasing order of priority are the following:

- Software trace

  The software can write directly to any of the 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.
- Hardware trace

  The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.
- Local timestamping

  The ITM contains a 21-bit counter clocked by the (predivided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

For more information on the ITM and how to use it, refer to *[4]*.

60.7.1 ITM registers

The ITM registers are located at the address range 0xE000 0000 to 0xE000 0FFC.

ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when writing

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	rw	rw

Bits 31:0 **STIMULUS[31:0]**: trace output data
write data is output on the trace bus as a software event packet.

ITM stimulus register x [alternate] (ITM_STIMRx)

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when reading

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DISABLE	FIFO_READY
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DISABLE**: Disable flag
0: stimulus port and ITM enabled
1: stimulus port and ITM disabled

Bit 0 **FIFO_READY**: FIFO ready indicator
0: stimulus port buffer is full (or port is disabled)
1: stimulus port can accept new write data

ITM trace enable register (ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMENA[31:0]**: stimulus port enable
Each bit x(0 to 31) enables the stimulus port associated with the ITM_STIMRx register.
0: port disabled
1: port enabled

ITM trace privilege register (ITM_TPR)

Address offset: 0xE40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: disable unprivileged access to ITM stimulus ports
Each bit controls eight stimulus ports.
XXX0: unprivileged access permitted on ports 0 to 7
XXX1: only privileged access permitted on ports 0 to 7
XX0X: unprivileged access permitted on ports 8 to 15
XX1X: only privileged access permitted on ports 8 to 15
X0XX: unprivileged access permitted on ports 16 to 23
X1XX: only privileged access permitted on ports 16 to 23
0XXX: unprivileged access permitted on ports 24 to 31
1XXX: only privileged access permitted on ports 24 to 31

ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								r	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
						rw	rw			rw	r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: indicates whether the ITM is currently processing events
0: not busy
1: busy

Bits 22:16 **TRACEBUSID[6:0]**: identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field.
Note: Different identifiers must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]**: local timestamp prescaler, used with the trace packet reference clock
0x0: no prescaling
0x1: Divide by 4.
0x2: Divide by 16.
0x3: Divide by 64.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STALLENA**: stall enable
0: Drop hardware source packets and generate an overflow if the ITM output is stalled.
1: Stall the processor to guarantee delivery of data trace packets.

Bit 4 **SWOENA**: SWO enable
Enables asynchronous clocking of the timestamp counter (read only).
0: Timestamp counter uses processor clock.

Bit 3 **TXENA**: transmit enable
Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: disabled
1: enabled

Bit 2 **SYNCENA**: synchronization packet transmission enable
The debugger setting this bit must also configure the DWT_CTRLR.SYNCTAP field for the correct synchronization speed.
0: disabled
1: enabled

Bit 1 **TSENA**: local timestamp generation enable
0: disabled
1: enabled

Bit 0 **ITMENA**: ITM enable
0: disabled
1: enabled

ITM device type architecture register (ITM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: ITM architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: ITM architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA01: ITM architecture

ITM device type register (ITM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: sub-type
0x4: associated with a bus, stimulus derived from bus activity

Bits 3:0 **MAJOR[3:0]**: major type
0x3: trace source

ITM CoreSight™ peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ITM CoreSight™ peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: ITM part number

ITM CoreSight™ peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: ITM part number

ITM CoreSight™ peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

ITM CoreSight™ peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

ITM CoreSight™ component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

ITM CoreSight™ peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

ITM CoreSight™ component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

ITM CoreSight™ component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

60.7.2 ITM register map

The ITM registers are located at the address range 0xE000 0000 to 0xE000 0FFC.

Table 723. ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 to 0x07C	**ITM_STIMR0 to ITM_STIMR31**	STIMULUS[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x07C to 0xDFC	Reserved	Reserved																															
0xE00	**ITM_TER**	STIMENA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04 to 0xE3C	Reserved	Reserved																															
0xE40	**ITM_TPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK [3:0]			
	Reset value																													0	0	0	0
0xE44 to 0xE7C	Reserved	Reserved																															
0xE80	**ITM_TCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE [1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0							0	0			0	0	0	0	0	0
0xE84 to 0xFB8	Reserved	Reserved																															

Table 723. ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFBC	**ITM_DEVARCHR**	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[3:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1
0xFC0 to 0xFC8	Reserved	Reserved																															
0xFCC	**ITM_DEVTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0xFD0	**ITM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	**ITM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	**ITM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	**ITM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	**ITM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**ITM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**ITM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	**ITM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**ITM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

60.8 Breakpoint unit (BPU)

The BPU allows the user to set hardware breakpoints. It contains eight comparators that monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint instruction.

For more information on the breakpoint unit and how to use it, refer to *[4]*.

60.8.1 BPU registers

The BPU registers are located at the address range 0xE0002000 to 0xE0002FFC.

BPU control register (BPU_CTRLR)

Address offset: 0x000

Reset value: 0x1000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			Res.	Res.	Res.	Res.	NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r					r	r	r	r			rw	rw

Bits 31:28 **REV[3:0]**: revision number
0x1: BPU version 2

Bits 27:15 Reserved, must be kept at reset value.

Bits 14:12, 7:4 **NUM_CODE[6:0]**: number of instruction address comparators supported
0x08: 8 instruction comparators supported

Bits 11:8, 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: Write protect key
A write to BPU_CTRLR register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: BPU enable
0: disabled
1: enabled

BPU comparator x register (BPU_COMPxR)

Address offset: 0x008 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPADDR[15:1]															BE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:1 **BPADDR[31:1]**: breakpoint address

Bit 0 **BE**: breakpoint enable
0: disabled
1: enabled

BPU device type architecture register (BPU_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DEVARCH register present
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: BPU architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: BPU architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA03: BPU architecture

BPU device type register (BPU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: sub-type
0x0: other

Bits 3:0 **MAJOR[3:0]**: major type
0x0: miscellaneous

BPU CoreSight™ peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

BPU CoreSight™ peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: BPU part number

BPU CoreSight™ peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: BPU part number

BPU CoreSight™ peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm ® JEDEC code

BPU CoreSight™ peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

BPU CoreSight™ component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

BPU CoreSight™ peripheral identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

BPU CoreSight™ component identity register 2 (BPU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

BPU CoreSight™ component identity register 3 (BPU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

60.8.2 BPU register map

The BPU registers are located at the address range 0xE000 2000 to 0xE000 2FFC.

Table 724. BPU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**BPU_CTRLR**	REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE [6:4]			Res.	Res.	Res.	Res.	NUM_CODE [3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1														0	0	0					1	0	0	0			0	0
0x004	Reserved	Reserved																															
0x008 to 0x024	**BPU_COMP0R to BPU_COMP7R**	BPADDR[31:1]																															BE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028 to 0xFB8	Reserved	Reserved																															
0xFBC	**BPU_DEVARCHR**	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1
0xFC0 to 0xFC8	Reserved	Reserved																															
0xFCC	**BPU_DEVTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	**BPU_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	**BPU_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	**BPU_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	**BPU_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	**BPU_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**BPU_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**BPU_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0

Table 724. BPU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	**BPU_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**BPU_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 718: Processor ROM table* for register boundary addresses.

60.9 Embedded Trace Macrocell (ETM)

The ETM is a CoreSight™ component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex®-M33 core to be traced. In the STM32G56x/55x/54x/53x, the ETM is configured for instruction trace only. Data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- Number of instructions executed in the same cycle
- Changes in program flow
- Current processor instruction state
- Addresses of memory locations accessed by load and store instructions
- Type, direction, and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM-M33 Technical Reference Manual *[6]*.

60.9.1 ETM registers

The ETM registers are located at the address range 0xE004 1000 to 0xE004 1FFC.

ETM programming control register (ETM_PRGCTLR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: trace unit enable
0: disabled
1: enabled

ETM status register (ETM_STATR)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: stability status
Indicates that the ETM-M33 registers are stable and can be read.
0: not stable
1: stable

Bit 0 **IDLE**: trace unit status
Indicates that the trace unit is inactive.
0: not idle
1: idle

ETM configuration register (ETM_CONFIGR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
			rw		rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RS**: return stack enable
0: disabled
1: enabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:5 **COND[5:0]**: conditional instruction tracing
0x0: conditional instruction tracing disabled
0x1: conditional load instructions traced
0x2: conditional store instructions traced
0x3: conditional load and store instructions traced
0x7: All conditional instructions traced

Bit 4 **CCI**: cycle counting in instruction trace
0: disabled
1: enabled

Bit 3 **BB**: branch broadcast mode
0: disabled
1: enabled

Bits 2:0 Reserved, must be kept at reset value.

ETM event control 0 register (ETM_EVENTCTL0R)

Address offset: 0x020

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
rw				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: resource type for event1
0: single selected resource
1: boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL1[3:0]**: resource number based on TYPE1
Selects the resource number, based on the value of TYPE1.
When TYPE1 = 0, a single resource from 0-15 defined by SEL1[3:0] is selected.
When TYPE1 = 1, a boolean combined resource pair defined by SEL1[2:0] is selected.

Bit 7 **TYPE0**: resource type for event0
0: single selected resource
1: boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: resource number based on TYPE0
Selects the resource number, based on the value of TYPE0.
When TYPE0 = 0, a single resource from 0-15 defined by SEL0[3:0] is selected.
When TYPE0 = 1, a boolean combined resource pair defined by SEL0[2:0] is selected.

ETM event control 1 register (ETM_EVENTCTL1R)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
			rw	rw										rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: low-power state behavior override
0: normal low-power state behavior
1: The resources and event trace generation are not affected by entry to a low-power state.

Bit 11 **ATB**: ATB trigger enable
0: disabled
1: enabled

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **INSTEN[1:0]**: instruction event generation
Enables generation of an event element in the instruction stream.
0bX0: Event0 does not cause an event element.
0bX1: Event0 causes an event element when it occurs.
0b0X: Event1 does not cause an event element.
0b1X: Event1 causes an event element when it occurs.

ETM stall control register (ETM_STALLCTLR)

Address offset: 0x02C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
					rw		rw					rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **INSTPRIORITY**: instruction trace priority
Prioritizes instruction trace if instruction trace buffer space is less than LEVEL[3:0].
0: The ETM must not prioritize instruction trace.
1: The ETM can prioritize instruction trace.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ISTALL**: processor stalling
Stalls processor based on instruction trace buffer space.
0: The ETM must not stall the processor.
1: The ETM can stall the processor.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LEVEL[3:0]**: Threshold at which stalling becomes active
This field provides four levels. This level can be varied to optimize the level of invasion caused by stalling, balanced against the risk of a FIFO overflow.
0x0: zero invasion, but greater risk of FIFO overflow
...
0xF: maximum invasion but less risk of FIFO overflow

ETM synchronization period register (ETM_SYNCPR)

Address offset: 0x034

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: synchronization period
Defines the number of bytes of trace between trace synchronization requests as a total of the number of bytes generated by the instruction stream.
0xA: 1024 bytes

ETM cycle count control register (ETM_CCCTLR)

Address offset: 0x038

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **THRESHOLD[11:0]**: instruction trace cycle count threshold

Sets the threshold value for instruction trace cycle counting. The threshold represents the minimum interval between cycle-count trace packets.

ETM trace identification register (ETM_TRACEIDR)

Address offset: 0x040

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace identification to output onto the trace bus

This field must be programmed with a unique value to differentiate it from other trace sources in the system.

Values 0x00 and 0x70-0x7F are reserved.

ETM ViewInst main control register (ETM_VICTLR)

Address offset: 0x080

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: exception level in secure state

Controls whether instruction tracing is enabled for the corresponding exception level, in secure state.

0bXXX0: instruction trace not generated in secure state, for exception level 0

0bXXX1: instruction trace generated in secure state, for exception level 0

0b0XXX: instruction trace not generated in secure state, for exception level 3

0b1XXX: instruction trace generated in secure state, for exception level 3

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: trace system error exception

0: The system error exception is traced only if the instruction or exception immediately before the system error exception is traced.

1: The system error exception is always traced.

Bit 10 **TRCRESET**: trace reset exception

0: The reset exception is traced only if the instruction or exception immediately before the reset exception is traced.

1: The reset exception is always traced.

Bit 9 **SSSTATUS**: start/stop logic status

0: stopped

1: started

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **EVENT[7:0]**: event selector

ETM counter reload value register 0 (ETM_CNTRLDVR0)

Address offset: 0x140

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: counter reload value

This value is loaded in to the counter each time the reload event occurs.

ETM identification register 8 (ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAXSPEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXSPEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MAXSPEC[31:0]**: maximum speculation depth

Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.

0x0: The maximum trace speculation depth is zero.

ETM identification register 9 (ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: number of P0 right-hand keys used

0x0: no P0 right-hand keys used in instruction trace

ETM identification register 10 (ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: number of P1 right-hand keys used (including normal and special keys)

0x0: no P1 right-hand keys used in instruction trace

ETM identification register 11 (ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: number of special P1 right-hand keys used
0x0: no special P1 right-hand keys used in any configuration

ETM identification register 12 (ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDKEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDKEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDKEY[31:0]**: number of conditional instruction right-hand keys used (including normal and special keys)
0x1: one conditional instruction right-hand key implemented

ETM identification register 13 (ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDSPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDSPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDSPC[31:0]**: number of special conditional instruction right-hand keys used
0x0: no special conditional instruction right-hand keys implemented

ETM implementation specific register 0 (ETM_IMSPECR0)

Address offset: 0x1C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: implementation specific extension support
0x0: no implementation specific extensions are supported

ETM identification register 0 (ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x2800 06E1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1]
		r												r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
r		r	r	r	r	r		r	r	r	r	r	r	r	

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: commit field meaning
Indicates the meaning of the commit field in some packets.
1: commit mode 1

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **TRCEXDATA**: trace data transfers for exceptions
Indicates support for the tracing of data transfers for exceptions and exception returns.
0: not implemented

Bits 16:15 **QSUPP[1:0]**: Q element support
0: not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: conditional results tracing
Indicates how conditional results are traced.
0: The trace unit indicates only if a conditional instruction passes or fails its condition code check

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported
0x1: two events

Bit 9 **RETSTACK**: return stack support
1: two entry return stacks

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: cycle counting support
1: cycle counting implemented

Bit 6 **TRCCOND**: conditional instruction support
1: conditional instruction tracing implemented

Bit 5 **TRCBB**: branch broadcast support
1: branch broadcast tracing implemented

Bits 4:3 **TRCDATA[1:0]**: data tracing support
0x0: data tracing not supported

Bits 2:1 **INSTP0[1:0]**: support for tracing of load and store instructions as P0 elements
0x0: not supported

Bit 0 Reserved, must be kept at reset value.

ETM identification register 1 (ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F421

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: trace unit designer
0x41: Arm®

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: major trace unit architecture version number
0x4: ETMv4

Bits 7:4 **TRCARCHMIN[3:0]**: minor trace unit architecture version number
0x2: minor revision 2

Bits 3:0 **REVISION[3:0]**: implementation revision number
0x1: implementation revision 1

ETM identification register 2 (ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: cycle counter size
0x0: 12 bits

Bits 24:20 **DVSIZE[4:0]**: data value size
0x0: data value size not supported

Bits 19:15 **DASIZE[4:0]**: data address size.
0x0: data address size not supported

Bits 14:10 **VMIDSIZE[4:0]**: virtual machine ID size
0x0: virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: context ID size
0x0: context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: instruction address size
0x4: maximum 32-bit address size

ETM identification register 3 (ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0F09 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: ETM_STALLCTLR.NOOVERFLOW implementation
0: not implemented

Bits 30:28 **NUMPROC[2:0]**: number of processors available for tracing
0x0: one processor

Bit 27 **SYSSTALL**: system support for stall control of the processor
1: system supports stall control

Bit 26 **STALLCTL**: stall control support
1: ETM_STALLCTLR implemented

Bit 25 **SYNCPR**: trace synchronization period support
1: ETM_SYNCPR is read-only for instruction trace only configuration. The trace synchronization period is fixed.

Bit 24 **TRCERR**: ETM_VICTLR.TRCERR implementation
0x1: implemented

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: privilege levels implementation
0x9: privilege levels thread and handler implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: minimum value that can be programmed to TRCCCCTLR.THRESHOLD
Defines the minimum cycle counting threshold.
0x4: minimum of four-instruction trace cycles

ETM identification register 4 (ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0011 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: number of virtual machine ID (VMID) comparators
0x0: VMID comparators not implemented

Bits 27:24 **NUMCIDC[3:0]**: number of context ID comparators
0x0: context ID comparators not supported

Bits 23:20 **NUMSSCC[3:0]**: number of single-shot comparator controls
0x1: one single-shot comparator control implemented

Bits 19:16 **NUMRSPAIR[3:0]**: number of resource selection pairs
0x1: two resource selection pairs implemented

Bits 15:12 **NUMPC[3:0]**: number of processor comparator inputs for the DWT
0x4: four processor comparator inputs implemented

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPDAC**: data address comparisons
0: data address comparisons not supported

Bits 7:4 **NUMDVC[3:0]**: number of data value comparators
0x0: no data value comparators implemented

Bits 3:0 **NUMACPAIRS[3:0]**: number of address comparator pairs
0x0: no address comparator pairs implemented

ETM identification register 5 (ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
r	r	r	r	r	r	r		r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: reduced function counter
1: counter 0 implemented as a reduced function counter

Bits 30:28 **NUMCNTR[2:0]**: number of counters
0x1: one counter implemented.

Bits 27:25 **NUMSEQSTATE[2:0]**: number of sequencer states
0x0: no sequencer states implemented.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: low-power state override support
1: low-power state override support implemented

Bit 22 **ATBTRIG**: ATB trigger support
1: ATB trigger support implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: number of bits of trace identification
0x7: 7-bit trace identification implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: number of external input selectors
0x0: no external input selectors implemented.

Bits 8:0 **NUMEXTIN[8:0]**: number of external inputs
0x004: four external inputs implemented.

ETM resource register 2 (ETM_RSCTLR2)

Address offset: 0x208

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: result of a combined pair of resources inversion
0: not inverted
1: inverted

Bit 20 **INV**: selected resources inversion
0: not inverted
1: inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: group of resources selection
0x0: external input selectors (select 0-3)
0x1: inputs from processor DWT comparators element (select 0-3)
0x2: counter at zero (select 0)
0x3: single-shot comparator (select 0)
Others: reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: more resources selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM resource register 3 (ETM_RSCTLR3)

Address offset: 0x20C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]		
											rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: selected resources inversion
0: not inverted
1: inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: group of resources selection
0x0: external input selectors (select 0-3)
0x1: inputs from processor DWT comparators element (select 0-3)
0x2: counter at zero (select 0)
0x3: single-shot comparator (select 0)
Others: reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: more resources selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM single-shot comparator control register 0 (ETM_SSCCR0)

Address offset: 0x280

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RST**: single-shot comparator reset
Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.
1: reset enabled

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (ETM_SSCSR0)

Address offset: 0x2A0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
												r	r	r	r

Bit 31 **STATUS**: single-shot comparator status
Indicates whether any of the selected comparators have matched.
0: no match occurred
1: at least one match occurred

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **PC**: processor comparator input sensitivity
1: single-shot comparator sensitive to processor comparator inputs

Bit 2 **DV**: data value comparator support
0: single-shot data value comparisons not supported

Bit 1 **DA**: data address comparator support
0: single-shot data address comparisons not supported

Bit 0 **INST**: instruction address comparator support
0: single-shot instruction address comparisons not supported

ETM single-shot processor comparator input control register 0 (ETM_SSPCICR0)

Address offset: 0x2C0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PC[3:0]**: processor comparator inputs selection for single-shot control
0xXXX0: processor comparator input 0 not selected
0xXXX1: processor comparator input 0 selected
0xXX0X: Processor comparator input 1 not selected
0xXX1X: processor comparator input 1 selected
0xX0XX: processor comparator input 2 not selected
0xX1XX: processor comparator input 2 selected
0x0XXX: processor comparator input 3 not selected
0x1XXX: processor comparator input 3 selected

ETM power-down control register (ETM_PDCR)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: power-up request
- 0: power-up not requested
- 1: power-up requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (ETM_PDSR)

Address offset: 0x314

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: sticky power-down state
- 0: Trace register power has not been removed since the ETM_PDSR was last read.
- 1: Trace register power has been removed since the ETM_PDSR was last read.

Bit 0 **POWER**: ETM power-up status
- 1: ETM powered up

ETM claim tag set register (ETM_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: claim tag bits setting

Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: claim tag bits reset

Write:
0000: no effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

ETM authentication status register (ETM_AUTHSTATR)

Address offset: 0xFB8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x2: secure non-invasive debug disabled
0x3: secure non-invasive debug enabled

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x2: non-secure non-invasive debug disabled
0x3: non-secure non-invasive debug enabled

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x0: not implemented

ETM device type architecture register (ETM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4772 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x2: ETM architecture v4.2

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x4: ETM architecture v4.2

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA13: ETM architecture

ETM CoreSight™ device type register (ETM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: device sub-type identifier
0x1: processor trace

Bits 3:0 **MAJORTYPE[3:0]**: device main type identifier
0x3: trace source

ETM CoreSight™ peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETM CoreSight™ peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: ETM part number

ETM CoreSight™ peripheral identity register 1 (ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: ETM part number

ETM CoreSight™ peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

ETM CoreSight™ peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

ETM CoreSight™ component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

ETM CoreSight™ peripheral identity register 1 (ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: trace generator component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

ETM CoreSight™ component identity register 2 (ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

ETM CoreSight™ component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

60.9.2 ETM register map

The ETM registers are accessed by the debugger at the address range 0xE0041000 to 0xE0041FFC.

Table 725. ETM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	**ETM_PRGCTLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x008	Reserved	Reserved																															
0x00C	**ETM_STATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
	Reset value																															X	X
0x010	**ETM_CONFIGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
	Reset value																				X		X	X	X	X	X	X	X	X			
0x014 to 0x01C	Reserved	Reserved																															
0x020	**ETM_ EVENTCTL0R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
	Reset value																	X				X	X	X	X	X				X	X	X	X
0x024	**ETM_ EVENTCTL1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
	Reset value																				X	X										X	X
0x028	Reserved	Reserved																															

Table 725. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	**ETM_STALLCTLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
	Reset value																						X		X					X	X	X	X
0x030	Reserved	Reserved																															
0x034	**ETM_SYNCPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]			
	Reset value																													0	1	0	1	0
0x038	**ETM_CCCTLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value																					X	X	X	X	X	X	X	X	X	X	X	X
0x03C	Reserved	Reserved																															
0x040	**ETM_TRACEIDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value																										X	X	X	X	X	X	X
0x044 to 0x07C	Reserved	Reserved																															
0x080	**ETM_VICTLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
	Reset value													X	X	X	X					X	X	X		X	X	X	X	X	X	X	X
0x084 to 0x13C	Reserved	Reserved																															
0x140	**ETM_CNTRLDVR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x144 to 0x17C	Reserved	Reserved																															
0x180	**ETM_IDR8**	MAXSPEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	**ETM_IDR9**	NUMP0KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	**ETM_IDR10**	NUMP1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	**ETM_IDR11**	NUMP1SPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	**ETM_IDR12**	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	**ETM_IDR13**	NUMCONDSPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198 to 0x1BC	Reserved	Reserved																															
0x1C0	**ETM_IMSPECR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT [3:0]			
	Reset value																													0	0	0	0
0x1C4 to 0x1DC	Reserved	Reserved																															

Table 725. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1E0	**ETM_IDR0**	Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1:0]		Res.	CONDTYPE [1:0]		NUMEVENT [1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
	Reset value			1												0	0	0		0	0	0	1	1		1	1	1	0	0	0	0	
0x1E4	**ETM_IDR1**	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ [3:0]				TRCARCHMIN [3:0]			REVISION [3:0]				
	Reset value	0	1	0	0	0	0	0	1													0	1	0	0	0	0	1	0	0	0	0	1
0x1E8	**ETM_IDR2**	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]					VMIDSIZE[4:0]			CIDSIZE[4:0]				IASIZE[4:0]							
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1EC	**ETM_IDR3**	NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	CCITMIN[11:0]											
	Reset value	0	0	0	0	1	1	1	1					1	0	0	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F0	**ETM_IDR4**	NUMVMIDC [3:0]				NUMCIDC [3:0]				NUMSSCC [3:0]				NUMRSPAIR [3:0]				NUMPC [3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC [3:0]				NUMACPAIRS [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0				0	0	0	0	0	0	0	0	0
0x1F4	**ETM_IDR5**	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE [2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE [5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL [2:0]			NUMEXTIN[8:0]								
	Reset value	1	0	0	1	0	0	0		1	1	0	0	0	1	1	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F8 to 0x204	Reserved	Reserved																															
0x208	**ETM_RSCTLR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value											X	X		X	X	X									X	X	X	X	X	X	X	X
0x20C	**ETM_RSCTLR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value												X		X	X	X									X	X	X	X	X	X	X	X
0x210 to 0x27C	Reserved	Reserved																															
0x280	**ETM_SSCCR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								X																								
0x284 to 0x29C	Reserved	Reserved																															
0x2A0	**ETM_SSCSR0**	STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
	Reset value	X																												X	X	X	X
0x2A4 to 0x2BC	Reserved	Reserved																															

Table 725. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C0	**ETM_SSPCICR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
	Reset value																													X	X	X	X
0x2C4 to 0x30C	Reserved	Reserved																															
0x310	**ETM_PDCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	
	Reset value																													0			
0x314	**ETM_PDSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value																															1	1
0x318 to 0xF9C	Reserved	Reserved																															
0xFA0	**ETM_CLAIMSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	**ETM_CLAIMCLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFB4	Reserved	Reserved																															
0xFB8	**ETM_AUTHSTATR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									X	X	X	X	X	X	X	X
0xFBC	**ETM_DEVARCHR**	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0to 0xFC8	Reserved	Reserved																															
0xFCC	**ETM_DEVTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYP [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0xFD0	**ETM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4to 0xFDC	Reserved	Reserved																															
0xFE0	**ETM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	**ETM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	**ETM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	**ETM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 725. ETM register map and reset values (continued)

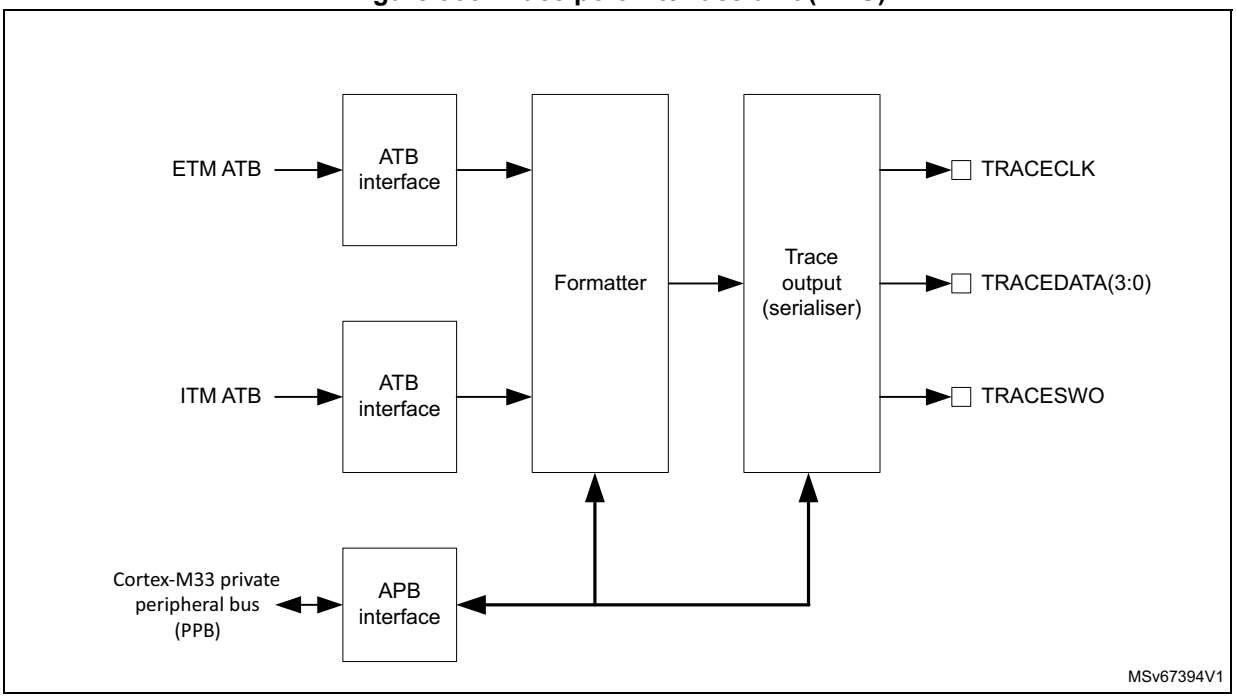
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	**ETM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**ETM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	**ETM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**ETM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 718: Processor ROM table* for register boundary addresses.

60.10 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in the figure below, the TPIU has two ATB slave ports for incoming trace data from the ETM and ITM respectively. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and four data outputs, TRACEDATA(3:0). The trace port width is programmable in the range 1, 2 or 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes, at the expense of bandwidth restriction of the trace port, and hence of the quantity of trace information that can be output in real time.

Figure 966. Trace port interface unit (TPIU)



Trace data can also be output on the serial-wire output, TRACESWO.

For more information on the trace port interface in the Cortex®-M33, refer to the Arm® Cortex®-M33 Technical Reference Manual *[5]*.

60.10.1 TPIU registers

TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: trace port sizes, from 1 to 32 pins

Bit n-1 when set, indicates that port size n is supported.

0x0000 000B: port sizes 1, 2 and 4 supported

TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: current trace port size

Bit n-1 when set, indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1, 2 or 4). Only one bit can be set, or unpredictable behavior may result.

This register must only be modified when the formatter is stopped.

TPIU asynchronous clock prescaler register (TPIU_ACPR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: baud rate for the asynchronous output, TRACESWO
The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]**: protocol used for trace output
0x0: parallel trace port mode
0x1: asynchronous SWO using Manchester encoding
0x2: asynchronous SWO using NRZ encoding
0x3: reserved

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: formatter stop

Indicates whether formatter can be stopped or not.

1: The formatter cannot be stopped.

Bit 2 **TCPRESENT**: TRACECTL output pin availability

Indicates whether the optional TRACECTL output pin is available for use.

0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: formatter stop

The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.

0: The formatter has not stopped.

Bit 0 **FLINPROG**: flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.

0: no flush in progress

1: flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
							r		rw					rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TRIGIN**: trigger on trigger in

1: Indicates a trigger in the trace stream when the TRIGIN input is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: flush on manual

0: flush completed

1: Generates a flush.

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** continuous formatting enable

Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output, ETM trace is discarded.

0: continuous formatting disabled

1: continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

TPIU periodic synchronization counter register (TPIU_PSCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PSCOUNT[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PSCOUNT[12:0]**: formatter frames counter

Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word synchronization frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: claim tag bits setting
Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: claim tag bits reset
Write:
0000: no effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

TPIU device configuration register (TPIU_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0000 0CA1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Serial-wire output, NRZ support
0x1: supported

Bit 10 **SWOMAN**: Serial-wire output, Manchester encoded format, support
0x1: supported

Bit 9 **TCLKDATA**: trace clock plus data support
0x0: supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relationship (synchronous or asynchronous)
0x1: asynchronous

Bits 4:0 **MAXNUM[4:0]**: number/type of ATB input port multiplexing
0x1: two input ports

TPIU device type identifier register (TPIU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: trace port component

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x1: trace sink component

TPIU CoreSight™ peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TPIU CoreSight™ peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: TPIU part number

TPIU CoreSight™ peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: TPIU part number

TPIU CoreSight™ peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

TPIU CoreSight™ peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

TPIU CoreSight CoreSight™ component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

TPIU CoreSight™ peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

TPIU CoreSight CoreSight™ component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

TPIU CoreSight CoreSight™ component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

60.10.2 TPIU register map

Table 726. TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TPIU_SSPSR**	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
0x004	**TPIU_CSPSR**	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008	Reserved	Reserved																															
0x010	**TPIU_ACPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x014 to 0x0EC	Reserved	Reserved																															
0x0F0	**TPIU_SPPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE [1:0]	
	Reset value																															0	1
0x0F4 to 0x2FC	Reserved	Reserved																															
0x300	**TPIU_FFSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304	**TPIU_FFCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
	Reset value																								1		0					1	

Table 726. TPIU register map and reset values (continued)

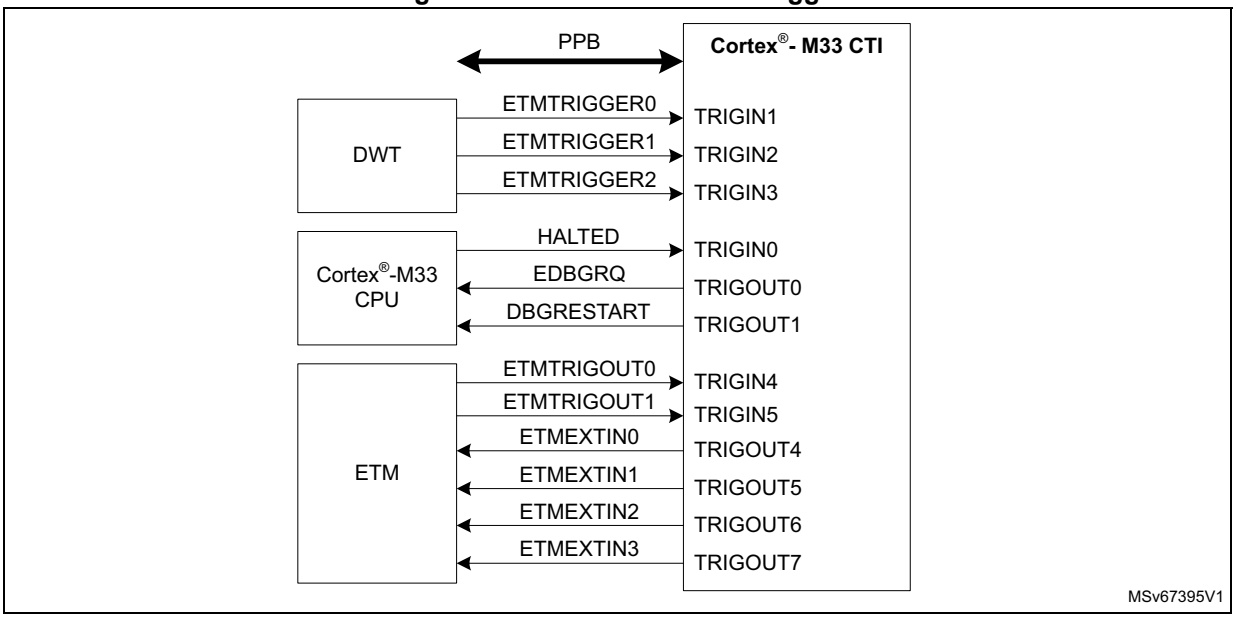
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x308	**TPIU_PSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
030C to 0xF9C	Reserved	Reserved																															
0xFA0	**TPIU_CLAIMSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	**TPIU_CLAIMCLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0FA8 to 0xFC4	Reserved	Reserved																															
0xFC8	**TPIU_DEVIDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ:	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MaXNUM[4:0]				
	Reset value																					1	1	0	0	1	0	1	0	0	0	0	1
0xFCC	**TPIU_DEVTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	**TPIU_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	**TPIU_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	**TPIU_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	**TPIU_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	**TPIU_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**TPIU_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**TPIU_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	**TPIU_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**TPIU_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 717: MCU ROM table* for register boundary addresses.

60.11 Cross-trigger interface (CTI)

The CTI (see *Figure 967*) allows cross triggering between the processor and the ETM.

Figure 967. Embedded cross trigger



The CTI enables events from various sources to trigger debug and/or trace activity. For example, a watchpoint reached in the processor can start or stop the code trace, or a trace comparator can halt the processor.

The trigger input and output signals for the CTI are listed in the tables below.

Table 727. CTI inputs

Number	Source signal	Source component	Comments
0	HALTED	CPU	Processor halted - CPU is in debug mode
1	ETMTRIGGER0	DWT	DWT comparator output 0
2	ETMTRIGGER1	DWT	DWT comparator output 1
3	ETMTRIGGER2	DWT	DWT comparator output 2
4	ETMTRIGOUT0	ETM	ETM event output 0
5	ETMTRIGOUT1	ETM	ETM event output 1
6	-	-	Not used
7	-	-	Not used

Table 728. CTI outputs

Number	Source signal	Destination component	Comments
0	EDBGRQ	CPU	CPU halt request - Puts CPU in debug mode
1	DBGRESTART	CPU	CPU restart request - CPU exits debug mode

Table 728. CTI outputs (continued)

Number	Source signal	Destination component	Comments
2	ETMEXTIN0	ETM	ETM event input 0
3	ETMEXTIN1	ETM	ETM event input 1
4	ETMEXTIN2	ETM	ETM event input 2
5	ETMEXTIN3	ETM	ETM event input 3
6	-	-	Not used
7	-	-	Not used

For more information on the cross-trigger interface CoreSight™ component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

60.11.1 CTI registers

The register file base address for the CTI is 0xE004 2000.

CTI control register (CTI_CONTROLR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: global CTI enable
0: disabled
1: enabled

CTI trigger acknowledge register (CTI_INTACKR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: trigger acknowledge

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSETR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: channel event setting
Read:
XXX0: channel 0 event inactive
XXX0: channel 0 event active
XX0X: channel 1 event inactive
XX1X: channel 1 event active
X0XX: channel 2 event inactive
X1XX: channel 2 event active
0XXX: channel 3 event inactive
1XXX: channel 3 event active
Write:
XXX0: no effect
XXX0: Sets event on channel 0.
XX0X: no effect
XX1X: Sets event on channel 1.
X0XX: no effect
X1XX: Sets event on channel 2.
0XXX: no effect
1XXX: Sets event on channel 3.

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: channel event clear
0000: no effect
XXX1: Clears event on channel 0.
XX1X: Clears event on channel 1.
X1XX: Clears event on channel 2.
1XXX: Clears event on channel 3.

CTI application pulse register (CTI_APPPULSER)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: pulse channel event
This register clears itself immediately.
0000: no effect
XXX1: Generates pulse on channel 0.
XX1X: Generates pulse on channel 1.
X1XX: Generates pulse on channel 2.
1XXX: Generates pulse on channel 3.

CTI trigger input x enable register (CTI_INENxR)

Address offset: 0x020 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: trigger input event enable

Enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0000: Trigger does not generate events on channels.

XXX1: Trigger x generates events on channel 0.

XX1X: Trigger x generates events on channel 1.

X1XX: Trigger x generates events on channel 2.

1XXX: Trigger x generates events on channel 3.

CTI trigger output x enable register (CTI_OUTENxR)

Address offset: 0x0A0 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: trigger output event enable

For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0000: Channel events do not generate triggers on trigger outputs.

XXX1: Channel 0 events generate triggers on trigger output x.

XX1X: Channel 1 events generate triggers on trigger output x.

X1XX: Channel 2 events generate triggers on trigger output x.

1XXX: Channel 3 events generate triggers on trigger output x.

CTI trigger input status register (CTI_TRGISTSR)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: trigger input status

There is one bit of the register for each CTITRIGINx input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger output status register (CTI_TRGOSTSR)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]**: trigger output status

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel input status register (CTI_CHINSTSR)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: channel input status

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel output status register (CTI_CHOUTSTSR)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: channel output status

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATER)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: channel output enable

For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.

0000: Channels events do not propagate.

XXX1: Channel 0 events propagate.

XX1X: Channel 1 events propagate.

X1XX: Channel 2 events propagate.

1XXX: Channel 3 events propagate.

CTI device configuration register (CTI_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: number of ECT channels available
0x4: four channels

Bits 15:8 **NUMTRIG[7:0]**: number of ECT triggers available
0x8: height trigger inputs and height trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: number of trigger input/output multiplexers
0x0: none

CTI device type identifier register (CTI_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight™ SoC-400 system.

CTI CoreSight™ peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CTI CoreSight™ peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: CTI part number

CTI CoreSight™ peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: CTI part number

CTI CoreSight™ peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CTI CoreSight™ peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

CTI CoreSight CoreSight™ component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

CTI CoreSight™ peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

CTI CoreSight CoreSight™ component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

CTI CoreSight CoreSight™ component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

60.11.2 CTI register map

Table 729. CTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**CTI_CONTROLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x004 to 0x00C	Reserved	Reserved																															
0x010	**CTI_INTACKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	**CTI_APPSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0
0x018	**CTI_APPCLEAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR [3:0]			
	Reset value																													0	0	0	0

Table 729. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	**CTI_APPPULSER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE [3:0]			
	Reset value																													X	X	X	X
0x020 to 0x03C	**CTI_INEN0R to CTI_INEN7R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN [3:0]			
	Reset value																													0	0	0	0
0x040 to 0x09C	Reserved	Reserved																															
0x0A0 to 0x0BC	**CTI_OUTEN0R to CTI_OUTEN7R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN [3:0]			
	Reset value																													0	0	0	0
0x0C0 to 0x12C	Reserved	Reserved																															
0x130	**CTI_TRGISTSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	**CTI_TRGOSTSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x138	**CTI_CHINSTSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS [3:0]			
	Reset value																													0	0	0	0
0x13C	**CTI_CHOUTSTSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
	Reset value																													0	0	0	0
0x140	**CTI_GATER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value																													1	1	1	1
0x144 to 0xFC4	Reserved	Reserved																															
0xFC8	**CTI_DEVIDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM [4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0
0xFCC	**CTI_DEVTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	**CTI_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	**CTI_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	0	0	1
0xFE4	**CTI_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1

Table 729. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE8	**CTI_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	**CTI_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**CTI_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**CTI_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	**CTI_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**CTI_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 718: Processor ROM table* for register boundary addresses.

60.12 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the software) to:

- Maintain the clock and power to the processor cores when in low-power modes (Sleep, Stop, or Standby).
- Maintain the clock and power to the system debug and trace components when in Low-power modes.
- Stop the clock to certain peripherals (SMBUS timeout, watchdogs, timers, RTC) when either processor core is stopped in debug mode.

60.12.1 Device ID

The DBGMCU includes an identity code register, DBGMCU_IDCODE. This register contains the ID code for the device. Debug tools can locate this register via the CoreSight™ discovery procedure described in *Section 60.5*.

60.12.2 Low-power mode emulation

When the device enters either Stop mode (clocks are stopped) or Standby mode (core power is switched off), the debugger can no longer access the debug access port and loses the connection with the device. To avoid this, the debugger (or software) can set the DBG_STANDBY and/or DBG_STOP bits in the *DBGMCU configuration register (DBGMCU_CR)*. These bits, when set, maintain the clock and power to the processor while the device is in the corresponding low-power mode. The processor remains in Sleep mode, and exits the low-power mode in the normal way. However, peripheral devices continue to operate, so the device behavior may not be identical to that of the actual low-power mode.

60.12.3 Peripheral clock freeze

The DBGMCU peripheral clock freeze registers allow the suspension of the operation of certain peripherals in debug mode. The peripherals supporting this feature are listed in *Table 730*.

Table 730. Peripheral clock freeze control bits

Bus	Control register	Peripheral	Description
APB1	DBGMCU_APB1FZR	I3C1	I3C1 SCL stall timeout counter
		I2C2	I2C2 SMBUS timeout
		I2C1	I2C1 SMBUS timeout
		IWDG	Independent watchdog
		WWDG	Window watchdog
		TIM14	General purpose timer 14
		TIM13	General purpose timer 13
		TIM12	General purpose timer 12
		TIM11	General purpose timer 11
		TIM10	General purpose timer 10
		TIM9	General purpose timer 9
		TIM7	General purpose timer 7
		TIM6	General purpose timer 6
		TIM3	General purpose timer 3(1)
		TIM2	General purpose timer 2
APB2	DBGMCU_APB2FZR	TIM5	General purpose timer 5(1)
		TIM4	General purpose timer 4(1)
		TIM17	General purpose timer 17
		TIM16	General purpose timer 16
		TIM15	General purpose timer 15
		TIM8	General purpose timer 8
		TIM1	General purpose timer 1
		HRTIM1	High resolution timer 1
APB3	DBGMCU_APB3FZR	RTC	Real time clock
		LPTIM1	Low power timer 1
		I2C4	I2C4 SMBUS timeout(1)
		I2C3	I2C3 SMBUS timeout(1)
		TIM27	General purpose timer 27
		TIM26	General purpose timer 26
		TIM20	General purpose timer 20(1)

Table 730. Peripheral clock freeze control bits (continued)

Bus	Control register	Peripheral	Description
AHB1	DBGMCU_AHB1FZR	GPDMA2 0 to 7	General purpose DMA2 channels 0 to 7
		GPDMA1 0 to 7	General purpose DMA1 channels 0 to 7

1. Category 3 devices

Each peripheral unit or DMA channel has a corresponding control bit, DBG_xxx_STOP, where xxx is the acronym of the peripheral (or DMA channel). The control bits are organized in DBGMCU_zzzFZR registers, where zzz corresponds to the name of the bus (AHB or APB). For example, DBGMCU_APB1LFZR contains the control bits for peripherals on the APB1L bus.

The control bit, when set, causes the corresponding peripheral operation to be suspended when the CPU is stopped in debug (HALTED = 1), according to the table below:

Table 731. Peripheral behavior in debug mode

HALTED	DBG_xxx_STOP	Peripheral behaviour
0	X	The operation continues.
1	0	The operation continues.
1	1	The operation is suspended.

The accessibility of the bits DBG_xxx_STOP by the debugger depends on the state of the authentication signal spiden.

When spiden = 1 (secure privilege debug enabled), all bits can be modified and read by both debugger and software (secure or non-nonsecure).

When spiden = 0 (secure privilege debug disabled), only bits corresponding to non-nonsecure peripherals (or DMA channels) can be modified by the debugger or software. All bits can be read.

The status (secure or non-nonsecure) of a TrustZone®–aware peripheral or a DMA channel, is signaled to the DBGMCU by the peripheral.

60.12.4 DBGMCU registers

The DBGMCU registers are reset only by a power-on reset. They are accessible to the debugger via the AHB access port at base address 0xE00E 6000, and to software at base address 0x4402 6000 (secure address: 0x5402 6000).

Note: Some register bits are not available on certain devices (refer to the device datasheet for availability of the associated peripheral). These bits should be considered as reserved and set to their reset value.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x00

Reset value: 0xXXXX 6XXX

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Revision
A: 0x1000
Z: 0x1001

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device identification
0x44A: STM32G55x/56x
0x44B: STM32G53x/54x

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x04

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, debugger accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCRT
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE [1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
								rw	rw	rw	rw		rw	rw	

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DCRT**: Debug credentials reset type
This bit selects which type of reset is used to revoke the debug authentication credentials
0: System reset
1: Power reset

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **TRACE_MODE[1:0]**: trace pin assignment
0x0: trace pins assigned for asynchronous mode (TRACESWO)
0x1: trace pins assigned for synchronous mode with a port width of 1 (TRACECK, TRACED0)
0x2: trace pins assigned for synchronous mode with a port width of 2 ((TRACECK, TRACED0-1)
0x3: trace pins assigned for synchronous mode with a port width of 4 ((TRACECK, TRACED0-3)

Bit 5 **TRACE_EN**: trace port and clock enable.
This bit enables the trace port clock, TRACECK.
0: disabled
1: enabled

Bit 4 **TRACE_IOEN**: trace pin enable
0: disabled - trace pins not assigned
1: enabled - trace pins assigned according to the value of TRACE_MODE field

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Allows debug in Standby mode
0: normal operation
All clocks are disabled and the core powered down automatically in Standby mode.
1: automatic clock stop/power down disabled
All active clocks and oscillators continue to run during Standby mode, and the core supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 1 **DBG_STOP**: Allows debug in Stop mode
0: normal operation
All clocks are disabled automatically in Stop mode.
1: automatic clock stop disabled
All active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB1 peripheral freeze register (DBGMCU_APB1FZR)

Address offset: 0x08

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
								rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	Res.	DBG_TIM3_STOP	DBG_TIM2_STOP
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DBG_I3C1_STOP**: I3C1 SCL stall counter stop in debug

0: normal operation. I3C1 SCL stall timeout counter continues to operate while CPU is in debug mode.

1: stop in debug. I3C1 SCL stall timeout counter is frozen while CPU is in debug mode.

Bit 22 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug

0: normal operation. I2C2 SMBUS timeout continues to operate while CPU is in debug mode.

1: stop in debug. I2C2 SMBUS timeout is frozen while CPU is in debug mode.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug

0: normal operation. I2C1 SMBUS timeout continues to operate while CPU is in debug mode.

1: stop in debug. I2C1 SMBUS timeout is frozen while CPU is in debug mode.

Bits 20:14 Reserved, must be kept at reset value.

Bit 13 **DBG_IWDG_STOP**: IWDG stop in debug

0: normal operation. IWDG continues to operate while CPU is in debug mode.

1: stop in debug. IWDG is frozen while CPU is in debug mode.

Bit 12 **DBG_WWDG_STOP**: WWDG stop in debug

0: normal operation. WWDG continues to operate while CPU is in debug mode.

1: stop in debug. WWDG is frozen while CPU is in debug mode.

Bit 11 **DBG_TIM14_STOP**: TIM14 stop in debug

0: normal operation. TIM14 continues to operate while CPU is in debug mode.

1: stop in debug. TIM14 is frozen while CPU is in debug mode.

Bit 10 **DBG_TIM13_STOP**: TIM13 stop in debug

0: normal operation. TIM13 continues to operate while CPU is in debug mode.

1: stop in debug. TIM13 is frozen while CPU is in debug mode.

Bit 9 **DBG_TIM12_STOP**: TIM12 stop in debug

0: normal operation. TIM12 continues to operate while CPU is in debug mode.

1: stop in debug. TIM12 is frozen while CPU is in debug mode.

Bit 8 **DBG_TIM11_STOP**: TIM11 stop in debug
0: normal operation. TIM11 continues to operate while CPU is in debug mode.
1: stop in debug. TIM11 is frozen while CPU is in debug mode.

Bit 7 **DBG_TIM10_STOP**: TIM10 stop in debug
0: normal operation. TIM10 continues to operate while CPU is in debug mode.
1: stop in debug. TIM10 is frozen while CPU is in debug mode.

Bit 6 **DBG_TIM9_STOP**: TIM9 stop in debug
0: normal operation. TIM9 continues to operate while CPU is in debug mode.
1: stop in debug. TIM9 is frozen while CPU is in debug mode.

Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: normal operation. TIM7 continues to operate while CPU is in debug mode.
1: stop in debug. TIM7 is frozen while CPU is in debug mode.

Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: normal operation. TIM6 continues to operate while CPU is in debug mode.
1: stop in debug. TIM6 is frozen while CPU is in debug mode.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug
0: normal operation. TIM3 continues to operate while CPU is in debug mode.
1: stop in debug. TIM3 is frozen while CPU is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in debug
0: normal operation. TIM2 continues to operate while CPU is in debug mode.
1: stop in debug. TIM2 is frozen while CPU is in debug mode.

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x10

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DBG_TIM5_STOP	DBG_TIM4_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP
	rw	rw											rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_HRTIM1_STOP
		rw		rw											rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DBG_TIM5_STOP**: TIM5 stop in debug
0: normal operation. TIM5 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM5 is frozen while CPU is in debug mode.

Bit 29 **DBG_TIM4_STOP**: TIM4 stop in debug
0: normal operation. TIM4 continues to operate while CPU is in debug mode.
1: stop in debug. TIM4 is frozen while CPU is in debug mode.

Bits 28:19 Reserved, must be kept at reset value.

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in debug
0: normal operation. TIM17 continues to operate while CPU is in debug mode.
1: stop in debug. TIM17 is frozen while CPU is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in debug
0: normal operation. TIM16 continues to operate while CPU is in debug mode.
1: stop in debug. TIM16 is frozen while CPU is in debug mode.

Bit 16 **DBG_TIM15_STOP**: TIM15 stop in debug
0: normal operation. TIM15 continues to operate while CPU is in debug mode.
1: stop in debug. TIM15 is frozen while CPU is in debug mode.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **DBG_TIM8_STOP**: TIM8 stop in debug
0: normal operation. TIM8 continues to operate while CPU is in debug mode.
1: stop in debug. TIM8 is frozen while CPU is in debug mode.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in debug
0: normal operation. TIM1 continues to operate while CPU is in debug mode.
1: stop in debug. TIM1 is frozen while CPU is in debug mode.

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **DBG_HRTIM1_STOP**: HRTIM1 stop in debug
0: normal operation. HRTIM1 continues to operate while CPU is in debug mode.
1: stop in debug. HRTIM1 is frozen while CPU is in debug mode.

DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZR)

Address offset: 0x14

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM1_STOP	Res.
	rw													rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DBG_I2C4_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM27_STOP	DBG_TIM26_STOP	DBG_TIM20_STOP	Res.	Res.
				rw	rw						rw	rw	rw		

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DBG_RTC_STOP**: RTC stop in debug
0: normal operation. RTC continues to operate while CPU is in debug mode.
1: stop in debug. RTC is frozen while CPU is in debug mode.

Bits 29:18 Reserved, must be kept at reset value.

Bit 17 **DBG_LPTIM1_STOP**: LPTIM1 stop in debug
0: normal operation. LPTIM1 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM1 is frozen while CPU is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DBG_I2C4_STOP**: I2C4 SMBUS timeout stop in debug
0: normal operation. I2C4 SMBUS timeout counter continues to operate while CPU is in debug mode.
1: stop in debug. I2C4 SMBUS timeout counter is frozen while CPU is in debug mode.

Bit 10 **DBG_I2C3_STOP**: I2C3 SMBUS timeout stop in debug
0: normal operation. I2C3 SMBUS timeout counter continues to operate while CPU is in debug mode.
1: stop in debug. I2C3 SMBUS timeout counter is frozen while CPU is in debug mode.

Bits 9:5 Reserved, must be kept at reset value.

Bit 4 **DBG_TIM27_STOP**: TIM27 stop in debug
0: normal operation. TIM27 continues to operate while CPU is in debug mode.
1: stop in debug. TIM27 is frozen while CPU is in debug mode.

Bit 3 **DBG_TIM26_STOP**: TIM26 stop in debug
0: normal operation. TIM26 continues to operate while CPU is in debug mode.
1: stop in debug. TIM26 is frozen while CPU is in debug mode.

Bit 2 **DBG_TIM20_STOP**: TIM20 stop in debug
0: normal operation. TIM20 continues to operate while CPU is in debug mode.
1: stop in debug. TIM20 is frozen while CPU is in debug mode.

Bits 1:0 Reserved, must be kept at reset value.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

Address offset: 0x20

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA2_7_STOP	DBG_GPDMA2_6_STOP	DBG_GPDMA2_5_STOP	DBG_GPDMA2_4_STOP	DBG_GPDMA2_3_STOP	DBG_GPDMA2_2_STOP	DBG_GPDMA2_1_STOP	DBG_GPDMA2_0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA1_7_STOP	DBG_GPDMA1_6_STOP	DBG_GPDMA1_5_STOP	DBG_GPDMA1_4_STOP	DBG_GPDMA1_3_STOP	DBG_GPDMA1_2_STOP	DBG_GPDMA1_1_STOP	DBG_GPDMA1_0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DBG_GPDMA2_7_STOP**: GPDMA2 channel 7 stop in debug
0: normal operation. GPDMA2 channel 7 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 7 is frozen while CPU is in debug mode.

Bit 22 **DBG_GPDMA2_6_STOP**: GPDMA2 channel 6 stop in debug
0: normal operation. GPDMA2 channel 6 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 6 is frozen while CPU is in debug mode.

Bit 21 **DBG_GPDMA2_5_STOP**: GPDMA2 channel 5 stop in debug
0: normal operation. GPDMA2 channel 5 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 5 is frozen while CPU is in debug mode.

Bit 20 **DBG_GPDMA2_4_STOP**: GPDMA2 channel 4 stop in debug
0: normal operation. GPDMA2 channel 4 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 4 is frozen while CPU is in debug mode.

Bit 19 **DBG_GPDMA2_3_STOP**: GPDMA2 channel 3 stop in debug
0: normal operation. GPDMA2 channel 3 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 3 is frozen while CPU is in debug mode.

Bit 18 **DBG_GPDMA2_2_STOP**: GPDMA2 channel 2 stop in debug
0: normal operation. GPDMA2 channel 2 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 2 is frozen while CPU is in debug mode.

Bit 17 **DBG_GPDMA2_1_STOP**: GPDMA2 channel 1 stop in debug
0: normal operation. GPDMA2 channel 1 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 1 is frozen while CPU is in debug mode.

Bit 16 **DBG_GPDMA2_0_STOP**: GPDMA2 channel 0 stop in debug
0: normal operation. GPDMA2 channel 0 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 0 is frozen while CPU is in debug mode.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DBG_GPDMA1_7_STOP**: GPDMA1 channel 7 stop in debug
0: normal operation. GPDMA1 channel 7 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 7 is frozen while CPU is in debug mode.

Bit 6 **DBG_GPDMA1_6_STOP**: GPDMA1 channel 6 stop in debug
0: normal operation. GPDMA1 channel 6 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 6 is frozen while CPU is in debug mode.

Bit 5 **DBG_GPDMA1_5_STOP**: GPDMA1 channel 5 stop in debug
0: normal operation. GPDMA1 channel 5 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 5 is frozen while CPU is in debug mode.

Bit 4 **DBG_GPDMA1_4_STOP**: GPDMA1 channel 4 stop in debug
0: normal operation. GPDMA1 channel 4 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 4 is frozen while CPU is in debug mode.

Bit 3 **DBG_GPDMA1_3_STOP**: GPDMA1 channel 3 stop in debug
0: normal operation. GPDMA1 channel 3 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 3 is frozen while CPU is in debug mode.

Bit 2 **DBG_GPDMA1_2_STOP**: GPDMA1 channel 2 stop in debug
0: normal operation. GPDMA1 channel 2 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 2 is frozen while CPU is in debug mode.

Bit 1 **DBG_GPDMA1_1_STOP**: GPDMA1 channel 1 stop in debug
0: normal operation. GPDMA1 channel 1 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 1 is frozen while CPU is in debug mode.

Bit 0 **DBG_GPDMA1_0_STOP**: GPDMA1 channel 0 stop in debug
0: normal operation. GPDMA1 channel 0 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 0 is frozen while CPU is in debug mode.

DBGMCU EDBGRQ debug request control (DBGMCU_EDBGRQ)

Address offset: 0x80

Reset value: 0x0000 0000

This register can be written and read by an external debugger while the system reset is asserted, or if debug is enabled after a successful authentication.

This register is read-only when accessed by the CPU, writes have no effect. Read access only returns meaningful data if debug is enabled after a successful authentication.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_CPU
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DBG_CPU**: drives the CPU external debug request (EDBGRQ) signal.
This bit can be used to halt the CPU in debug after reset is released (vector catch), instead of writing to VC_CORERESET in DEMCR register.
0: External debug request de-asserted.
1: External debug request asserted.

DBGMCU status register (DBGMCU_SR)

Address offset: 0xFC

Reset value: 0x0001 XX03

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AP_ENABLED[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP_PRESENT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port AP n is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)
Bit n = 0: APn locked
Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port AP n is present in device
Bit n = 0: APn absent
Bit n = 1: APn present

DBGMCU debug authentication mailbox host register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0xXXXX XXXX

This register is read only when accessed by the CPU, writes have no effect.

This register can be written and read by an external debugger when system reset is asserted, or when access is granted during the authentication process.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Debug host to device mailbox message.
The debug host requests authentication by writing a value to this register prior to releasing the system reset. During debug authentication the debug host communicates with the device (CPU) via this register.

DBGMCU debug authentication mailbox device register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

This register is read only when accessed via the debug port, writes have no effect.

This register can be written and read by the CPU.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Device to debug host mailbox message.
During debug authentication the device (CPU) communicates with the debug host via this register.

DBGMCU debug authentication mailbox acknowledge register (DBGMCU_DBG_AUTH_ACK)

Address offset: 0x108

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ACK	HOST_ACK
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DEV_ACK**: Device to host acknowledge.

This bit is set by hardware when the device (CPU) writes a message in the DBGMCU_DBG_AUTH_DEVICE register. It is reset automatically when the host (debugger) reads the message.

0: DBGMCU_DBG_AUTH_DEVICE register is empty

1. DBGMCU_DBG_AUTH_DEVICE register contains an unread message

Bit 0 **HOST_ACK**: Host to device acknowledge.

This bit is set by hardware when the host (debugger) writes a message in the DBGMCU_DBG_AUTH_HOST register. It is reset automatically when the device (CPU) reads the message.

0: DBGMCU_DBG_AUTH_HOST register is empty

1. DBGMCU_DBG_AUTH_HOST register contains an unread message

DBGMCU CoreSight™ peripheral identity register 4 (DBGMCU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC code

DBGMCU CoreSight™ peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x00: DBGMCU part number

DBGMCU CoreSight™ peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

This register is always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x0: DBGMCU part number

DBGMCU CoreSight™ peripheral identity register 2 (DBGMCU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

DBGMCU CoreSight™ peripheral identity register 3 (DBGMCU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

DBGMCU CoreSight CoreSight™ component identity register 0 (DBGMCU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

DBGMCU CoreSight CoreSight™ component identity register 1 (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0xF: Non-CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

DBGMCU CoreSight CoreSight™ component identity register 2 (DBGMCU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

DBGMCU CoreSight CoreSight™ component identity register 3 (DBGMCU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

60.12.5 DBGMCU register map

Table 732. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DBGMCU_IDCODE**	REV_ID[15:0]																Res	Res	Res	Res	DEV_ID[11:0]											
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x004	**DBGMCU_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE [1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
	Reset value																0									0	0	0	0		0	0	

Table 732. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	**DBGMCU_APB1FZR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	Res.	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value									0	0	0								0	0	0	0	0	0	0	0	0	0			0	0
0x0C0	Reserved	Reserved																															
0x010	**DBGMCU_APB2FZR**	Res.	DBG_TIM5_STOP	DBG_TIM4_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_HRTIM1_STOP
	Reset value		0	0											0	0	0			0		0											0
0x04	**DBGMCU_APB3FZR**	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	DBG_I2C4_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM27_STOP	DBG_TIM26_STOP	DBG_TIM20_STOP	Res.	Res.
	Reset value		0													0						0	0						0	0	0		
0x018 to 0x01C	Reserved	Reserved																															
0x020	**DBGMCU_AHB1FZR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA2_7_STOP	DBG_GPDMA2_6_STOP	DBG_GPDMA2_5_STOP	DBG_GPDMA2_4_STOP	DBG_GPDMA2_3_STOP	DBG_GPDMA2_2_STOP	DBG_GPDMA2_1_STOP	DBG_GPDMA2_0_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA1_7_STOP	DBG_GPDMA1_6_STOP	DBG_GPDMA1_5_STOP	DBG_GPDMA1_4_STOP	DBG_GPDMA1_3_STOP	DBG_GPDMA1_2_STOP	DBG_GPDMA1_1_STOP	DBG_GPDMA1_0_STOP
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x024 to 0x07C	Reserved	Reserved																															
0x80	**DBGMCU_EDBGRQ**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CPU_DBG
	Reset value																																0
0x084 to 0x0F8	Reserved	Reserved																															
0x0FC	**DBGMCU_SR**	AP_ENABLED[15:0]																AP_PRESENT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
0x100	**DBGMCU_DBG_AUTH_HOST**	MESSAGE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x104	**DBGMCU_DBG_AUTH_DEVICE**	MESSAGE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 732. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x108	**DBGMCU_DBG_AUTH_ACK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ACK	HOST_ACK
	Reset value																															0	0
0x10C to 0xFBC	Reserved	Reserved																															
0xFD0	**DBGMCU_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	**DBGMCU_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFE4	**DBGMCU_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	0	0	0	0
0xFE8	**DBGMCU_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	**DBGMCU_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**DBGMCU_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**DBGMCU_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	**DBGMCU_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**DBGMCU_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

60.13 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8 Aug 2013
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16 March 2015
3. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
4. DDI 0553A (ID092917) - Arm® v8-M Architecture Reference Manual, Issue A.f, 29 September 2017
5. 100230_0002_00_en - Arm® Cortex®-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017
6. 100232_0001_00_en - Arm® CoreSight™ ETM-M33 r0p1 Technical Reference Manual, Issue 0001-00, 3 February 2017

61 Device electronic signature

The device electronic signature is stored in the system memory area of the flash memory module and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the devices.

61.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- For use as serial numbers (for example USB string serial numbers or other end applications)
- For use as part of the security keys in order to increase the security of code in flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal flash memory
- To activate secure boot processes

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x08FF F800

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]:** X and Y coordinates on the wafer

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **UID[63:40]:** LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** LOT_NUM[55:24]
Lot number (ASCII encoded)

61.2 Flash size data register

Base address: 0x08FF F80C

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size
This field indicates the size of the device flash memory expressed in Kbytes.
As an example, 0x800 corresponds to 2048 Kbytes.

61.3 Package data register

Base address: 0x08FF F80E

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[6:0]						
									r	r	r	r	r	r	r

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **PKG[6:0]**: Package type
0000000: UFQFPN32
0100100: UFQFPN40
0000001: LQFP48
0000010: TQFP48-EP
0000011: UFQFPN48
0000100: LQFP64
0000101: TQFP64-EP
0000110: LQFP80
0000111: LQFP80-EP
0001000: LQFP100
0001001: LQFP100-EP
0001010: LQFP128
0001011: LQFP128-EP
0001101: UFBGA121
0011010: LQFP128 SMPS
Others: reserved

Note: Refer to the product datasheet for availability of packages on a specific device.

61.4 Temperature calibration value

Base address: 0x08FF F810

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VREFINT_CAL [31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFINT_CAL [15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **VREFINT_CAL [31:0]**: Raw data acquired at 30°C, V_{DDA} = 3.3 V

61.5 Temperature calibration value low

Base address: 0x08FF F814

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS_CAL1 [31:16]															
R	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS_CAL1 [15:0]															
R	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TS_CAL1 [31:0]**: Temperature sensor raw data acquired value at 30°C, V_{DDA} = 3.3 V

61.6 Temperature calibration value high

Base address: 0x08FF F818

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS_CAL2 [31:16]															
R	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS_CAL2 [15:0]															
R	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TS_CAL2 [31:0]**: Temperature sensor raw data acquired value at 140°C, V_{DDA} = 3.3 V

61.7 PUF reference value (256bit)

Base address: 0x08FF F81C + (0x04 * x), (x =0...15)

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

A set of 16 registers containing the reference result for the PUF encryption.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUF_REF_Valuex[31:16]															
r	r	r	r	r	r	r	r	R	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUF_REF_Valuex[15:0]															
r	r	r	r	r	r	r	r	R	r	r	r	r	r	r	r

Bits 31:0 **PUF_REF_Valuex**: The x-th word of reference product of PUF_REF_Value = SAES(PUF, ClearPufRef). For detailed description of the PUF validation process refer to *Section 7: Embedded flash memory (FLASH)*.

62 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers must re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers must not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

63 Revision history

Table 733. Document revision history

Date	Revision	Changes
30-Oct-2026	1	Initial release.

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers’ market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved